Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Lampret

Avtomatska analiza in klasifikacija glasbenih besedil na podlagi čustev

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2017

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Lampret

Avtomatska analiza in klasifikacija glasbenih besedil na podlagi čustev

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Avtomatska analiza in klasifikacija glasbenih besedil na podlagi čustev

Tematika naloge:

V diplomski nalogi preučite stanje na področju avtomatske analize glasbenih besedil na podlagi čustev. Na podlagi izbranih del implementirajte lasten sistem za klasifikacijo in regresijo besedil glede na valenco in aktivnost čustev. Implementirajte obdelavo besedil, izračun ustreznih značilk in uporabite metode strojnega učenja za klasifikacijo in regresijo. Sistem evalvirajte na javnih podatkovnih zbirkah.

Zahvaljujem se mentorju za pomoč in potrpežljivost. Zahvaljujem se tudi vsem drugim, ki so mi v času študija stali ob strani in me podpirali.

Kazalo

Povzetek

Naslov: Avtomatska analiza in klasifikacija glasbenih besedil na podlagi čustev

Avtor: Rok Lampret

Pred kratkim so v študiji [46] ugotovili, da lahko z uporabo nekaterih novih značilk občutno izboljšajo modele za klasifikacijo in regresijo glasbenih besedil glede na valenco in aktivnost čustev. Na podlagi te študije smo implementirali svoj sistem za tovrstno avtomatsko analizo glasbenih besedil. Delali smo z glasbenimi besedili v angleškem jeziku. Najprej smo s spleta pridobili vsa glasbena besedila. Ta besedila smo z našim procesorjem besedil preoblikovali v primerno obliko za ekstrakcijo značilk. Pripravili smo vrsto funkcij za pridobivanje značilk, s katerimi smo nato iz besedil izluščili različne tipe značilk. Nato smo značilke, s pomočjo algoritmov za selekcijo značilk, razvrstili po pomembnosti in izbrali le najpomembnejše. Z naključnim iskanjem parametrov smo optimizirali parametre učnih algoritmov, ki smo jih učili na izbranih značilkah. Za učenje klasifikatorjev in regresorjev smo uporabili metodo podpornih vektorjev (angl. *Support Vector Machine*) in Gradient Boosting. Uspešnost naših modelov smo na koncu ocenili s stratificiranim 10-kratnim prečnim preverjanjem [37]. V delu predstavimo metode, ki smo jih uporabili za izgradnjo sistema, končno rešitev in rezultate, ki jih naš sistem dosega v primerjavi s sistemom iz prej omenjene študije.

Ključne besede: glasbena besedila, prepoznavanje čustev, pridobivanje značilk, procesiranje naravnega jezika, strojno učenje.

Abstract

Title: Automated Analysis and Emotion-based Classification of Music Lyrics

Author: Rok Lampret

In a recent study [46] on Lyrics Music Emotion Recognition a new set of features was proposed. These new features were proved to increase accuracy of existing models for classification and regression based on valence and arousal of emotion in lyrics. Based on the findings of this study we have implemented our own system for automatically acquiring, analyzing and classifying lyrics. We only dealt with lyrics in English. First we acquired the lyrics from the web. Then we prepared the lyrics for feature extraction, using the preprocessor we implemented. A variety of functions were implemented for feature extraction, which were then used to extract features from the preprocessed lyrics. Using feature selection algorithms we ranked the features and selected only the best. Using randomized hyper-parameter optimization we optimized the parameters of learning methods for our models. For classification and regression we used two learning algorithms, Support Vector Machine and Gradient Boosting. In the end we evaluated our models with stratified 10-fold cross validation [37]. In this work we present the methods that we used to build our system, final solution and the results that we achieved in comparison to the study we based our system on.

Keywords: emotion recognition, feature extraction, lyrics, machine learning, natural language processing.

Poglavje 1

Uvod

Glasba lahko vzbudi ali spremeni počutje in čustva [34, 35, 59]. Glasbo predvajajo oglaševalci, da spremenijo počutje potrošnikov [1, 7], psihoterapevti pa jo uporabljajo pri zdravljenju čustvenih motenj [23]. Študije kažejo, da se ob poslušanju glasbe aktivirajo podobni deli možganov za nagrajevanje, kot jih aktivirajo seks, hrana in droge [6, 48]. Vse kaže na to, da je pomemben razlog za popularnost glasbe skozi čas in prostor čustvena nagrada, ki jo nekdo prejme ob poslušanju glasbe.

Zato ni čudno, da se vse bolj za merilo pri iskanju glasbe uporabljajo tudi čustva in počutje. Velike glasbene podatkovne baze vse hitreje rastejo. Da so vsi vnosi v bazo pravilno kategorizirani, vzame veliko časa in ročnega dela. Poleg tega je kategoriziranje glasbe glede na prevladujoče čustvo oz. počutje zelo subjektivno in zelo odvisno od osebe, ki to delo opravlja. Rešitev za oba problema je sistem, ki tako delo avtomatizira [32]. Idealen sistem bi lahko na podlagi značilk iz zvočnega zapisa, besedila in drugih metapodatkov popolnoma avtomatsko in natančno kategoriziral pesmi.

V tem diplomskem delu smo se posvetili implementaciji svojega sistema za avtomatsko analizo in klasificiranje glasbenih besedil na podlagi čustev. Besedila smo klasificirali v kvadrante in poloble Russellovega krožnega modela [57]. Izvedli smo še regresijo besedil glede na vrednosti za valenco in aktivnost. Za učenje smo uporabili metodo podpornih vektorjev in Gradient

Boosting.

1.1 Vsebina diplomskega dela

Diplomsko delo vsebuje uvod, pet poglavij in zaključek. V uvodnem poglavju predstavimo tematiko dela. V poglavju 2 naredimo pregled področja prepoznavanja čustev. V poglavju 3 predstavimo problem. V poglavju 4 opišemo pristope k reševanju problema. V poglavju 5 predstavimo našo rešitev, od pridobivanja besedil do učenja modelov. V poglavju 6 predstavimo naše rezultate. V zaključku predstavimo glavna opažanja in predloge za izboljšave in nadaljnje delo.

Poglavje 2

Pregled področja

2.1 Teorija čustev

Merriam-Webster [19] definira čustvo kot „zavestno duševno reakcijo (kot npr. jeza ali strah), doživeto subjektivno kot močno občutenje navadno usmerjeno proti določenemu objektu, ki ga navadno spremljajo psihološke in čustvene spremembe v telesu“. Klasifikacija čustev je način, kako razločiti eno čustvo od drugega [20].

Glede na raziskave v psihologiji lahko razločimo tri pristope k modeliranju (predstavitvi) čustev: kategorični, dimenzionalni in komponentni pristop [24]. Vendar celo po več kot stoletju raziskav se raziskovalci na področju prepoznave čustev še vedno ne strinjajo, kateri pristop k čustvom je za kateri problem najboljši [25].

Kategorični modeli. Kategorični pristop temelji na študijah o osnovnih čustvih [14, 66, 67]. Takšen pristop pravi, da obstajajo osnovna čustva, ki so zakodirana v naše možgane in so univerzalna [18]. Ekman [17] navaja šest univerzalnih osnovnih čustev: veselje, žalost, presenečanje, strah, jeza in gnus.

Enega od najstarejših takšnih modelov je sestavil Havener [30], ki je 66 pridevnikov razvrstil v osem čustvenih skupin. Hu [31] je sestavil kategorični model s petimi čustvenimi kategorijami na podlagi oznak za razpoloženje

s spletnega vira AllMusic[1]. Ta model je prevzelo tudi združenje MIREX[2] (angl. *Music Information Retrieval Evaluation eXchange*) in je nasploh danes pogosto uporabljen model.

Dimenzionalni modeli. Dimenzionalni pristop pravi, da čustva niso popolnoma ločljiva, ampak so med seboj sistematično povezana. Pri takem pristopu so navadno čustva opisana v dvo- ali tridimenzionalnem prostoru. Dimenzije tega prostora so valenca, aktivnost in dominanca.

Valenca meri, kako prijetno se oseba počuti ob določenem čustvu [47]. Višja valenca pomeni bolj prijetno čustvo. Npr. jeza in strah sta neprijetni čustvi (nizka valenca), medtem ko je veselje prijetno čustvo (visoka valenca).

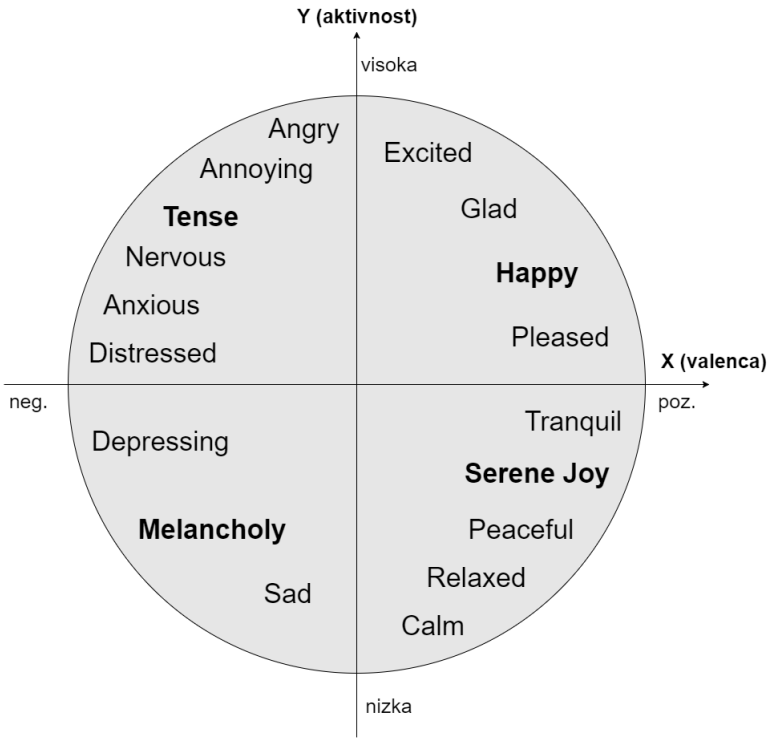
Aktivnost ne meri intenzitete čustva ampak kako aktivno oz. energično se oseba počuti ob določenem čustvu [47]. Npr. medtem ko sta jeza in dolgčas obe negativni čustvi, ima jeza veliko višjo aktivnost kot pa dolgčas.

Dominanca meri, kako zelo podrejeno se oseba počuti [47]. Nizka dominanca pomeni, da se oseba počuti podrejeno, visoka dominanca pomeni, da oseba čuti, da ima nadzor. Npr. oseba se pri jezi počuti nadrejeno, pri strahu pa podrejeno.

Eden od uveljavljenih modelov za dimenzionalno predstavitev čustev je Russellov krožni model [57] (angl. *Russell's circumplex model*). Čustva so predstavljena v dvodimenzionalnem VA prostoru, kjer x koordinata predstavlja valenco, tj. prijetnost (V) čustva in y koordinata predstavlja aktivnost čustva (A). V okviru tega dela smo implementirali sistem, ki glasbena besedila klasificira v kvadrante ali poloble, glede na valenco ali na aktivnost Rusellovega modela.

Komponentni modeli. Komponentni modeli čustev temeljijo na teoriji ocenjevanja [58] (angl. *appraisal theory*). Komponentni pristop k čustvom je nekakšen podaljšek dimenzionalnega pristopa in trdi, da se čustva tvorijo skozi nenehno, rekurzivno subjektivno ocenjevanje tako notranjega stanja osebe kot stanja zunanjega sveta. Tak pristop vidi čustva skozi spremembe

[1] `http://www.allmusic.com/`
[2] `http://www.music-ir.org/`



Slika 2.1: Russellov krožni model, povzeto po Yang [72]

v motivaciji, psiholoških reakcijah, motoričnem izražanju in občutkih osebe. Prednost takega pristopa je, da se ne omeji na nekaj fiksnih kategorij ali samo nekaj osnovnih dimenzij čustev. Vendar pa takšni modeli zahtevajo zelo kompleksno in prefinjeno merjenje sprememb in se zdi, da trenutno takšni modeli še niso primerni za uporabo na področju prepoznavanja čustev iz glasbe [25].

2.2 Prepoznavanje čustev

Avtomatsko prepoznavanje čustev je široko področje. Ukvarja se s prepoznavanjem čustev iz različnih vrst signalov:

- govora (npr. [2, 42, 52, 53, 61]),

- glasbe (npr. [9, 54, 70, 72]),
- obrazne mimike (npr. [11, 49, 62]),
- drže telesa (npr. [21]),
- besedila (npr. [44, 69])
- fizioloških signalov (npr. [26, 33]).

Veliko dela je bilo narejenega predvsem z bimodalnimi modeli za prepoznavanje čustev iz govora in obrazne mimike (npr. [12, 13, 15, 42, 73]). Področje prepoznavanja čustev združuje tehnike iz vrste drugih področij, kot so procesiranje signalov, računalniški vid in strojno učenje. V praksi se uporabljajo različni učni algoritmi, pogosti so: skriti model Markova (angl. *Hidden Markov Model, HMM*) (npr. [52]), metoda podpornih vektorjev (angl. *Support Vector Machine, SVM*) (npr. [42]), nevronske mreže (angl. *Neural network*) (npr. [28]), k-najbližjih sosedov (angl. *k-nearest neighbors, kNN*) (npr. [62]), mešanice Gaussovih porazdelitev (angl. *Gaussian Mixture Models, GMM*) (npr. [50]) in odločitvena drevesa (angl. *Decision trees*) (npr. [10]).

Prepoznavanje čustev ima potencial aplikacije na mnogih področjih. Nekatera področja so: inteligentni roboti in umetna inteligenca, „Text-to-Speech“ sistemi za čustveno branje besedil, komunikacija človek-računalnik, pametni avtomobili, video igre in e-šolstvo [38].

2.3 MIR

Pridobivanje informacij iz glasbe (angl. *Music Information Retrieval, MIR*) je široko interdisciplinarno področje, ki se ukvarja s pridobivanjem informacij iz glasbe. Združuje področja strojnega učenja, procesiranja signalov, psihologije in muzikologije. Področje se je začelo razvijati v osemdesetih letih prejšnjega stoletja, vendar več pozornosti prejema šele zadnji dve desetletji.

Cilji področja MIR so različni, nekateri so: razvoj priporočilnih sistemov, avtomatska transkripcija glasbe (npr. [4]), prepoznavanje instrumentov (npr. [29]), avtomatska kategorizacija glasbe in celo avtomatsko generiranje glasbe (npr. [51]). Za relevantno in nepristransko ovrednotenje teh opravil s področja MIR skrbi Pobuda o skupni evalvaciji pri pridobivanju informacij iz glasbe - MIREX. MIREX formalizira opravila s področja MIR in postopke za evalvacijo algoritmov, ki ta opravila rešujejo.

Problem avtomatske kategorizacije glasbe se ukvarja predvsem s kategorizacijo glede na glasbene zvrsti (žanre) [41] in glede na čustva (tudi razpoloženja) v glasbi [36]. Običajen postopek je, da se iz glasbenega posnetka, glasbenega besedila in/ali drugih metapodatkov pridobijo značilke, s katerimi se nato z algoritmi strojnega učenja zgradijo klasifikatorji. Za dober rezultat so pomembne predvsem tri stvari: dobre značilke, učni algoritem in glasbena zbirka (učna množica).

2.3.1 MER

Eno pomembnih področij MIR-a je prepoznavanje čustev v glasbi (angl. *Music Emotion Recognition, MER*). Sprva je MER temeljilo na prepoznavanju čustev iz zvočnih zapisov [44]. Sčasoma se je, zaradi semantičnega prepada med nizkonivojskimi značilkami zvoka in visokonivojskim razmišljanjem človeka, napredek na področju prepoznavanja čustev iz zvočnega zapisa ustavil in so začeli raziskovalci iskati nove rešitve. Pojavili so se bimodalni modeli, ki temeljijo na značilkah iz zvočnega zapisa in besedil [32, 40]. Takšni modeli dosegajo boljše rezultate kot pa ločeni modeli [45]. Za klasifikacijo glasbe glede na čustva iz avdio posnetkov obstaja več zbirk (npr. [9, 16, 22, 54, 60]). Za klasifikacijo na podlagi glasbenih besedil pa se zdi, da takšnih javno dostopnih zbirk primanjkuje. Zbirke, ki bi imela za angleška glasbena besedila podatke o valenci in aktivnosti, pa do nedavnega verjetno sploh ni bilo [46]. V nadaljevanju predstavimo nekaj relevantnih študij s področja prepoznavanja čustev iz glasbe.

Pri Laurier [40] in Cho [8] se je največji napredek, potem ko so avdio

značilkam dodali še značilke iz besedila, pokazal predvsem pri klasificiranju v kategorije „vesel“ in „žalosten“.

Han idr. [27] so razvili sistem za prepoznavanje čustev iz glasbenih posnetkov. V njihovem primeru se je najboljše izkazala klasifikacija na podlagi regresije s SVR (metoda podpornih vektorjev za regresijo) v polarnem koordinatnem sistemu. Klasificirali so v enajst čustvenih kategorij, definiranih na dvodimenzionalnem Thayerjevem modelu [65].

Song idr. [63] so ugotovili, da se za klasifikacijo glasbenih posnetkov, glede na čustva, bolje obnese SVM klasifikator s polinomskim jedrom kot pa z RBF jedrom.

Lin idr. [43] so uporabili podatek o zvrsti glasbe, da so izboljšali klasifikacijo glasbenih posnetkov glede na čustva. Za vsako glasbeno zvrst posebej so pripravili specifičen klasifikacijski model za čustva. Končni sistem je glasbeni posnetek najprej klasificiral v glasbeno zvrst in nato na podlagi glasbene zvrsti uporabil specifičen klasifikator za klasifikacijo v čustvene kategorije. Takšen dvoslojen model se je izkazal za bolj natančnega, kot samo enoslojen sistem.

Pred kratkim je Malheiro [46] ugotovil, da lahko z uporabo novih dodatnih značilk občutno izboljša obstoječe modele za klasifikacijo in regresijo glasbenih besedil. Nove značilke je uporabil pri regresiji glede na valenco in aktivnost in pri klasifikaciji glede na kvadrante in poloble Russelovega krožnega modela. V ta namen je pripravil tudi glasbeno zbirko 180 besedil, označenih z vrednostmi za valenco in aktivnost. Za testiranje je pripravil še 771 besedil, označenih glede na kvadrante. Poleg že uveljavljenih značilk vsebine, ki temeljijo na modelu vreče besed (angl. *Bag-of-Words model*), je preizkusil še vrsto novih značilk na podlagi strukture, stila in pomenskosti (semantike) besedila. Ta študija je tudi izhodišče za to diplomsko delo. Na podlagi ugotovitev te študije smo implementirali svoj sistem za klasifikacijo glasbenih besedil.

Poglavje 3

Predstavitev problema

Namen tega diplomskega dela je implementirati sistem za klasifikacijo glasbenih besedil v kvadrante (in poloble) Rusellovega modela in regresijo glede na valenco in aktivnost čustev. Sledili smo Malheiru [46], kar se da dosledno, da bi bila na koncu primerjava rezultatov čim bolj smiselna. V članku veliko stvari ni bilo opisanih do potankosti, zato smo morali nekoliko eksperimentirati, da smo prišli do podobnih rezultatov. Med drugim tudi nismo mogli uporabiti vseh orodij, ki jih je uporabil, saj so nekatera plačljiva, druga pa nedosegljiva na spletu.

Malheiro [46] je tudi pripravil javno dostopno zbirko ročno anotiranih glasbenih besedil. Sestavlja jo učna množica 180 besedil in testna množica 771 besedil. Besedila so bila skrbno izbrana, tako da je njihova porazdelitev po kvadrantih dokaj enakomerna. Besedila obsegajo različne glasbene žanre in obdobja. Samo učna množica pa vsebuje tudi podatke o čustveni valenci in aktivnosti besedil. Ker besedila sama niso mogla biti zbrana in objavljena, zaradi avtorskih pravic, so podani le naslovi spletnih strani s temi besedili. Tako je bil prvi korak pri izgradnji našega sistema pridobiti vsa potrebna besedila s spleta.

Sledilo je čiščenje besedil. Iz besedil smo odstranili vse, kar ni del dejanskega glasbenega besedila. Večina glasbenih besedil na spletu vsebuje še vrsto metapodatkov, kot so značke za posamezne kitice (npr. refren), po-

novitve vrstic in verzov in opisi glasbenega ozadja (odmevi, instrumenti). Velikokrat je na začetku ali na koncu besedila tudi še zapisan izvajalec in naslov pesmi, lahko tudi ime albuma in leto izdaje.

Potem ko smo imeli besedila očiščena, smo jih označili s POS oznakami (angl. *Part-of-Speech tag*) in besedila lematizirali. Na podlagi tega smo v naslednjem koraku pridobili značilke iz besedil. Te značilke so predstavljene na koncu tega poglavja.

Potem ko smo pridobili vse značilke iz besedil, smo lahko začeli z učenjem klasifikatorjev in regresorjev. Vse naučene modele smo nato ocenili. Iz posameznih najboljših modelov smo združili značilke, da smo dobili še kombinirane modele.

3.1 Značilke vsebine

Značilke vsebine (angl. *Content-Based Features*) so značilke, ki se navezujejo na vsebino besedila. To so najbolj splošne in pogosto uporabljene značilke v procesiranju naravnega jezika (angl. *Natural Language Processing*). Eden najpreprostejših modelov za predstavitev besedila z vsebinskimi značilkami je model vreče besed. Tudi za naš sistem je to osnovni model. Ta model je natančneje predstavljen v poglavju 4.2. Vendar pa se da, zaradi specifik glasbenih besedil, iz besedil izluščiti še več vrst drugih značilk.

Malheiro [46] navaja, da je v svoji študiji eksperimentiral z različnimi vrečami besed. Zgradil je unigrame, bigrame in trigrame iz prvotnih besedil in besedil, ki so bila prej podvržena krnjenju (angl. *stemming*) ali/in odstranjevanju prepovedanih besed (angl. *stop words removal*). Zgradil je še bido 5-grame iz POS oznak besedil. Za najboljše se jih je na koncu izkazalo naslednjih pet: unigrami z odstranjevanjem prepovedanih besed, unigrami z odstranjevanjem prepovedanih besed in krnjenjem, tf-idf bigrami z odstranjevanjem prepovedanih besed, bigrami in trigrami POS oznak. Mi smo ravno tako zgradili vse te vreče besed in naredili svoje poskuse z njimi.

3.2 Stilistične značilke

Stilistične značilke (angl. *Stylistic-Based Features*) se navezujejo na stil, v katerem je besedilo napisano, npr. kakšne besede so uporabljene v besedilu.

Malherio [46] je predlagal dva modela stilističnih značilk. Prvi model značilk je štetje besednih vrst (na podlagi POS oznak). Uporabil je 36 oznak besednih vrst iz projekta Penn Treebank [64]. Drugi model je sestavljen iz treh značilk: število slengovskih besed ali besednih zvez v besedilu, število besed, ki se začnejo z veliko začetnico (prva beseda v vrstici ne šteje), in število besed, ki so zapisane v celoti z velikimi črkami.

3.3 Značilke strukture pesmi

Značilke strukture pesmi (angl. *Song-Structure-Based Features*) so značilke, ki se navezujejo na sestavo pesmi. Malheiro [46] pravi, da najverjetneje te vrste značilk pred njim še niso bile uporabljene v LMER (angl. *Lyrics-based MER*).

Predlagane značilke so (verz tukaj pomeni vse kitice, ki niso refren): število ponovitev refrena, število ponovitev naslova pesmi, število vseh kitic (verzi in refreni), število verzov, ali se besedilo začne z refrenom, delež verzov od vseh kitic, delež refrenov od vseh kitic, ali se pesem konča s ponovitvijo vsaj dveh refrenov, ali se verzi in refreni v pesmi izmenjujejo (VRVR...), ali je med dvema verzoma vsaj dvakrat refren (VRRVRR...), ali je med dvema refrenoma vsaj en verz (VVRVR).

3.4 Semantične značilke

Semantične značilke (angl. *Semantic-Based Features*) so značilke, ki se navezujejo na semantiko besedila. Izkaže se, da so te značilke zelo pomembne in nosijo veliko pomena pri prepoznavanju čustev. Te značilke so najkompleksnejše, zato smo za pridobitev značilk te vrste uporabili že obstoječa orodja

in slovarje. To so: Synesketch[1], *General Inquirer*[2] (GI), *Dictionary of Affect in Language*[3] (DAL), *Affective Norms for English Words*[4] (ANEW) in Warrinerjev slovar [68]. Poleg teh orodij je Malheiro [46] uporabil še dva druga. Prvo je licenčni program *Language Inquiry and Word Count*[5] (LIWC). Drugo pa je ConceptNet[6]. Tega bi uporabili tudi mi, vendar pa nismo mogli prenesti prave verzije programa s spleta (obrazec na uradni strani projekta ne deluje več).

Značilke, ki jih iz besedil izlušči LIWC, so se v Malheirovi [46] študiji pokazale za zelo pomembne in se njihov primanjkljaj kaže v našem sistemu. LIWC na podlagi štetja besed skupaj s slovarjem, v katerem vsaka beseda spada v določene kategorije, izračuna določene metrike. Nekatere izmed metrik, ki jih LIWC izračuna, in kategorije besed[7] so:

- število vseh besed v besedilu
- jezikovne metrike (število besed na stavek, število besed z več kot šestimi črkami)
- slovnične metrike (pravilni glagoli, pridevniki, primerjalniki)
- štetje čustvenih besed (pozitivna čustva, anksioznost, jeza)
- štetje socialnih besed (družina, prijatelji)
- štetje besed, povezanih z biološkimi procesi (telo, spolnost, zdravje/bolezen)
- štetje besed, povezanih z relativnostjo (gibanje, prostor, čas)

[1] http://krcadinac.com/synesketch/
[2] http://www.wjh.harvard.edu/~inquirer
[3] https://www.god-helmet.com/wp/whissel-dictionary-of-affect/index.htm
[4] http://csea.phhp.ufl.edu/media/anewmessage.html
[5] http://www.liwc.net/
[6] http://alumni.media.mit.edu/~hugo/conceptnet/
[7] https://liwc.wpengine.com/compare-dictionaries/

- štejte besed, povezanih z osebnimi zadevami (delo, prosti čas, dom, denar, religija, smrt)
- štetje neformalnih besed (psovke, mašila, internetni sleng)

LIWC je več kot očitno zanimivo orodje za analizo besedila, predvsem iz psihološkega in socialnega vidika.

Na podoben način je zgrajen tudi slovar GI. Vsebuje skoraj dvanajst tisoč besed. Besede razvršča v 182 različnih kategorij, pri čemer vsaka beseda lahko pripada več kategorijam. Nekatere med njimi so „Positive“, „Negative“, „Strong“, „Hostile“, „Active“, „Passive“, „Aquatic“, „Casual“, „Fail“, „Goal“, „Legal“, „Object“, „Religion“, „Space“, „Time“, „Think“.

Slovarja DAL in ANEW vsebujeta podatke o čustveni valenci in aktivnosti besed. DAL ima poleg tega za besede še podatek „imagery“, ki pove, kako lahko si je v mislih ustvariti sliko besede. ANEW ima za besede še podatek o dominanci. Podoben je tudi Warrinerjev slovar [68], le da je obsežnejši. Vsebuje podatke o valenci, aktivnosti in dominanci za več kot trinajst tisoč angleških lem. DAL in ANEW skupaj vsebujeta nekaj več kot sedem tisoč različnih besed.

V tabeli 3.1 so primeri prvih treh besed z najnižjo in najvišjo valenco v slovarjih DAL in ANEW, v tabeli 3.2 pa za besede v slovarju Warriner [68]. Skala za valenco in aktivnost se med slovarji razlikuje, zato smo vse skale poenotili na interval $[1, 3]$. Takšno skalo uporablja DAL. Slovarja ANEW in Warriner [68] drugače uporabljata skalo od 1 do 9. Slovarji se med seboj razlikujejo ne samo po številu besed (DAL jih ima 8743, ANEW 1030 in Warriner [68] 13915), ampak tudi po „letu izdelave“. DAL je najstarejši, iz leta 1986, ANEW iz leta 1999 in najnovejši Warriner [68], iz leta 2013. Verjetno so tudi zaradi tega določene besede v različnih slovarjih različno klasificirane. V tabeli 3.3 je nekaj takih primerov besed, ki v vseh treh slovarjih spadajo v drug kvadrant.

Synesketch pa za besedilo izračuna osem uteži, šest za posamezna čustva (veselje, žalost, jeza, strah, gnus, presenečanje) in pa dve skupni uteži (splošna utež, valenca) glede čustev v besedilu.

beseda	val.	akt.
limitations	1.000	1.125
missing	1.000	1.125
shut	1.000	1.142
socialize	3.000	2.800
comedy	3.000	2.833
lover	3.000	3.000

beseda	val.	akt.
suicide	1.063	2.183
rape	1.063	2.453
funeral	1.098	1.985
paradise	2.930	2.030
love	2.930	2.360
triumphant	2.955	2.445

Tabela 3.1: Primeri besed z najnižjo in najvišjo valenco iz slovarjev DAL (levo) in ANEW (desno).

beseda	valenca	aktivnost
pedophile	1.065	2.013
rapist	1.075	2.333
AIDS	1.083	2.000
happy	2.868	2.263
happiness	2.870	2.375
vacation	2.883	2.055

Tabela 3.2: Primeri besed z najnižjo in najvišjo valenco iz slovarja Warriner [68].

	v. DAL	a. DAL	Q	v. ANEW	a. ANEW	Q	v. Warr.	a. Warr.	Q
bees	1.714	1.889	q3	1.550	2.378	q2	2.035	2.133	q1
leader	1.750	1.875	q3	2.658	2.318	q1	2.310	1.990	q4
boxer	1.500	2.778	q2	2.128	2.030	q1	2.053	1.785	q4
girl	1.833	1.875	q3	2.468	1.823	q4	2.538	2.058	q1
hard	1.286	2.429	q2	2.055	2.030	q1	1.838	1.875	q3
news	1.778	2.333	q2	2.075	2.043	q1	1.990	1.903	q3

Tabela 3.3: Primeri besed, ki v slovarjih DAL, ANEW in Warriner [68] spadajo v različne kvadrante.

Poglavje 4

Metode dela

V tem poglavju predstavimo pomembnejše metode in modele, ki smo jih uporabili za izgradnjo našega sistema. Naslednji opisi so površinski in so namenjeni zgolj razumevanju principov delovanja uporabljenih metod in modelov. Nobenega od naslednjih algoritmov nismo tudi dejansko implementirali sami, ampak smo uporabili že obstoječe rešitve iz različnih Python knjižnic. Na koncu vsakega opisa tudi piše, katero implementacijo smo uporabili.

4.1 POS označevanje

Vsaki besedi lahko določimo besedno vrsto. Nekatere besedne vrste v slovenščini so glagol, samostalnik, pridevnik in prislov. POS označevanje pomeni to, da vsaki besedi določimo oznako, ki predstavlja pripadajočo besedno vrsto besede. Na primer, angleški besedi „have“ pripada POS oznaka `VB`, ki predstavlja glagol v osnovni obliki. Besedi „had“ pa pripada POS oznaka `VBD`, ki predstavlja glagol v pretekli obliki. Ti dve oznaki sta fini POS oznaki. Grobi POS oznaki za obe besedi bi bila zgolj `VERB`, ki predstavlja vse glagole. POS razčlenjevalniki po navadi besedilo sami razbijejo v žetone, tj. osnovne elemente besedila. Poleg besed so to še ločila in števila. POS razčlenjevalniki potem določijo POS oznake posameznim žetonom. Za

to smo uporabili knjižnico Syntaxnet[1], ki vsebuje trenutno najnatančnejši razčlenjevalnik za angleška besedila Parsey McParseface [55].

4.2 Model vreče besed

Model vreče besed je način, kako lahko besedilo poenostavljeno predstavimo. Ta model je eden najpogosteje uporabljenih modelov za predstavitev besedila z značilkami ravno zaradi svoje preprostosti. Poleg uporabe v tekstovni analizi se uporablja tudi na področju računalniškega vida.

Vreča besed je neurejena množica besed, ki so si med seboj paroma različne in kjer za vsako besedo vemo število pojavitev v dokumentu. Beseda lahko tukaj pomeni črko, zlog, dejansko besedo, na področju računalniškega vida bi to lahko bilo zaporedje pik v sliki ipd. Dokument je lahko pred grajenjem vreče podvržen določenim transformacijam, najpogosteje krnjenju in odstranjevanju prepovedanih besed. Nato se vreča zgradi s štetjem vseh posameznih različnih besed v dokumentu. Tako se zgradi model vreče besed, ki pa je pravzaprav poseben primer n-gram modela, kjer je $n = 1$ (unigram). Lahko zgradimo tudi modele z bigrami, trigrami ali n-grami. Za grajenje modela z bigrami preprosto vzamemo po dve zaporedni besedi (bigram) v korpusu in potem v modelu to predstavlja eno besedo. Trigrami bi bili, če vzamemo po tri zaporedne besede in n-grami, če vzamemo po n zaporednih besed. Pri unigramih se izgubi prostorska informacija in lahko imata dva različna dokumenta popolnoma enaki vreči besed. Vzemimo na primer naslednji dve angleški povedi:

$$d_1 = \text{„John is better than Mary“}$$
$$d_2 = \text{„Mary is better than John“}$$

Oba dokumenta d_1 in d_2 bi imela enaki vreči besed, vendar pa sta povedi pomensko ravno obratni. Zato je uporaba n-gramov za $n > 1$ bolj primerna, kjer moramo ohraniti še nekaj prostorske informacije.

[1] `https://github.com/tensorflow/models/tree/master/syntaxnet`

V našem sistemu smo uporabili implementacijo modela vreče besed iz Python knjižnice scikit-learn[2]. V tej knjižnici je model vreče besed implementiran v razredu CountVectorizer. S CountVectorizerjem lahko zgradimo različne n-gram modele, lahko celo mešamo med sabo npr. model z unigrami in bigrami. CountVectorizerju lahko podamo svojo množico besed, za katere želimo, da jih šteje v besedilu. Lahko pa tudi sam zgradi besedišče iz danih dokumentov. Rezultat CountVectorizerjeve transformacije so vektorji pogostosti simbolov. CountVectorizer vsaki besedi v besedišču določi indeks, tako da je b_i i-ta beseda v besedišču. Tako i-ta komponenta v vektorju pogostosti simbolov predstavlja pogostost besede b_i v dokumentu.

4.3 Krnjenje in lematizacija

Krnjenje (tudi *korenjenje*) in lematizacija sta dva načina preoblikovanja besed v osnovno formo. Namen je več različnih besed preslikati v eno samo pomensko enako besedo. Na primer besede „hiša“, „hiše“, „hišam“ bi lahko preslikali v „hiša“. Osnova besede se pri krnjenju imenuje koren, pri lematizaciji pa lema. Da dobimo osnovo besede, se moramo znebiti artefaktov v besedah, ki se pojavijo npr. zaradi sklanjanja ali spreganja. V idealnem primeru bi besede pretvorili v slovarske oblike besed. V slovenščini je to za glagole nedoločnik, za samostalnike pa 1. oseba ednine v imenovalniku. To poskuša pravzaprav doseči lematizacija. S pomočjo morfološke analize stavka in slovarja poskuša za vsako besedo najti pomensko pravilno osnovno formo besede. Po drugi strani pa krnjenje poskuša pri vsaki posamezni besedi najti zgolj koren besede. Največkrat to pomeni, da besedi odreže končnice ali pa morda predpone, za katere ve, da so pogosto rezultat sklanjanja, spreganja in drugih transformacij. Tako bi recimo besedam „hiša“, „hiše“, „hišam“ lahko našli skupni koren „hiš“. Krnjenje ne vidi širšega konteksta in dela samo z eno besedo naenkrat, medtem ko lematizacija poskuša s pomočjo analize stavka ali dokumenta delovati bolj informirano. K analizi za lematizacijo

[2] `http://scikit-learn.org/stable/index.html`

spada tudi POS označevanje.

4.4 Izbira značilk z ReliefF

ReliefF [39] je algoritem za ocenjevanje pomembnosti značilk za klasifikacijske probleme. Algoritem ReliefF je nadgradnja algoritma Relief. Za regresijske probleme obstaja posebna različica ReliefF algoritma, ki se imenuje RReliefF.

Denimo, da imamo matriko X, ki predstavlja našo množico značilk. Vrstice v X predstavljajo učne primere, stolpci v X predstavljajo posamezne značilke. Za vsak učni primer v X vemo tudi njegov razred. Algoritem v grobem deluje tako, da si n-krat izbere po en naključni učni primer R v X. Za učni vzorec R najde k najbližjih (glede na Evklidsko razdaljo med vektorjema značilk) sosedov (vrstic v X), ki spadajo v isti razred kot R. Ti najbližji sosedje se imenujejo „nearest hit“. Potem za učni vzorec R najde še k najbližjih sosedov (učnih vzorcev v X), ki ne spadajo v isti razred kot R. Ti najbližji sosedje se imenujejo „nearest miss“. Če je število razredov več kot dva, potem algoritem najde k „nearest miss“ za vsak razred (drugačen od R) posebej. Nato sledi posodabljanje uteži. Vsaka značilka začne z utežjo 0,0. Po vsaki iteraciji se utež za značilko a_i posodobi tako, da se ji odšteje vsota razlik R-ja in vseh k „nearest hit“ v značilki a_i in prišteje vsota razlik R-ja in vseh k „nearest miss“ v značilki a_i. Če je razredov več, se vsota razlik med R in k „nearest miss“ izračuna za vsak razred posebej in se vsota razlik uteži z verjetnostjo razreda. Praktično to pomeni, da bolj kot je značilka različna znotraj istega razreda in bolj podobna med različnimi razredi, manjšo utež dobi. Obratno pa značilke, ki so si znotraj istega razreda bolj podobne in med različnimi razredi bolj različne, dobijo večjo utež. Rezultat algoritma so izračunane uteži za vsak atribut posebej. Iz uteži potem ni težko razbrati najboljše značilke.

V našem sistemu smo uporabili implementacijo algoritma ReliefF iz knjižnice scikit-feature[3]. Edini parameter, ki ga algoritmu podamo, je k. V

[3] `http://featureselection.asu.edu/`

praksi se izkaže, da je v splošnem $k = 10$ primerna vrednost [39].

4.5 Metode učenja

4.5.1 Metoda podpornih vektorjev

Metoda podpornih vektorjev je algoritem za nadzorovano strojno učenje. Lahko se uporablja za klasifikacijo in regresijo. Cilj algoritma je najti hiperravnino v n-dimenzionalnem prostoru, kjer je n število značilk, ki kar se da najbolje med seboj ločuje različne razrede. Na primer, če imamo učne primere v 2-dimenzionalnem prostoru, ločene v dva razreda, želimo poiskati premico, ki med seboj najbolje loči ta dva razreda. Točke (učni primeri), ki so najbližje tej premici, imenujemo podporni vektorji. Od tu tudi ime tej metodi. Premica, ki najbolje ločuje ta dva razreda, ima največji rob (angl. *margin*), to je razdalja od premice do podpornih vektorjev. Če bi katerega od podpornih vektorjev odstranili ali pa bi se kateri od podpornih vektorjev spremenil, bi se najverjetneje spremenila tudi ločevalna ravnina.

V primeru, ko razredi med sabo niso linearno ločljivi, uporabimo trik z jedrom (angl. *kerneling*). To pomeni, da učne primere preslikamo v višji dimenzionalni prostor in poskušamo najti ločevalno ravnino v tem višje dimenzijskem prostoru. Pogosti jedri, uporabljeni v praski, sta polinomski in RBF (angl. *Radial Basis Function*). Ta jedra imajo tudi vrsto parametrov. Parametre metode in jeder je treba previdno izbrati, saj lahko slaba izbira privede do preohlapnega ločevanja razredov ali pa do pretiranega prilagajanja učni množici (angl. *overfitting*). Pomembna parametra pri teh dveh jedrih sta C in *gama*. Oba kontrolirata, kako zelo naj se ločevalna črta prilagaja učnim podatkom in preveliki vrednosti za ta dva parametra lahko povzročita pretirano prilagajanje.

V našem sistemu smo uporabili implementaciji metode podpornih vektorjev za klasifikacijo in regresijo iz knjižnice scikit-learn, ki se imenujeta SVC in SVR.

4.5.2 Gradient Boosting

Gradient Boosting je ena izmed ansambelskih metod strojnega učenja. Lahko se uporablja za klasifikacijo in regresijo. Ansambelske metode naučijo več modelov (iz iste ali različnih vrst strojnega učenja), da na koncu skupaj dosežejo boljše rezultate, kot bi jih posamezni modeli. Metode tipa Boosting gradijo več šibkejših modelov, ki na koncu skupaj tvorijo en močnejši model. Šibkejši model tukaj pomeni tak model, ki je lahko le za malo boljši od naključnega modela. Po navadi gradijo šibka (plitva) odločitvena drevesa. Ansambel gradijo iterativno. V vsaki iteraciji zgradijo nov model, ki se osredotoči na klasificiranje problematičnih učnih vzorcev, ki jih modeli iz prejšnjih iteracij niso pravilno napovedali, in ga dodajo končni rešitvi. Tako poskušajo v vsaki iteraciji znižati skupno napako. To ponavljajo, dokler ni napaka dovolj nizka. Gradient Boosting računa napako z gradienti, od tu tudi ime.

Implementacija Gradient Boostinga v Pythonu, ki smo jo uporabili, se imenuje XGBoost[4]. Za preizkus tega algoritma in te specifične implementacije smo se odločili, ker je bil to zmagovalni algoritem na več tekmovanjih v zadnjem času[5]. Nastavljivih je vrsto parametrov, nekateri med njimi so *n_estimators*, *max_depth*, *subsample*, *gamma* in *learning_rate*. Kontrolirajo med drugim, koliko dreves zgraditi in do kakšne globine. Ponovno je treba paziti pri izbiri parametrov, da končni model ni pretirano prilagojen učni množici.

4.6 Prečno preverjanje

Pri računanju natančnosti modela je pomembno, da natančnosti ne ocenimo na isti množici, kot smo izvedli učenje modela. Ta ocena je največkrat pretirano optimistična. K učenju modela spadajo izbira najpomembnejših značilk,

[4] `https://xgboost.readthedocs.io/en/latest/`

[5] `https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions`

optimizacija hiperparametrov in dejansko učenje modela na podatkih. Da bi dobili bolj realno oceno modela, po navadi podatke razdelimo na učno in testno množico. Posledično se lahko zgodi, da je učnih primerov zalo malo. Tak problem rešujemo s k-kratnim prečnim preverjanjem (angl. *k-fold Cross Validation*). Deluje tako, da vse podatke razdelimo na k-enako velikih delov. V vsaki iteraciji izberemo $k-1$ delov in jih združimo. To postane učna množica. Ostane $\frac{1}{k}$ podatkov, ki jih vzamemo za testno množico. Potem model naučimo na učni množici in ocenimo natančnost na testni množici. To ponovimo k-krat, tako da je vsak od k delov enkrat testna množica. Na koncu za oceno vzamemo povprečje vseh izmerjenih ocen. Takšna ocena modela je bolj realna kot pa, če bi učenje in testiranje izvedli na celotni množici. Pri klasifikacijskih primerih, kjer vemo razrede primerov, lahko uporabimo stratificirano prečno preverjanje (angl. *Stratified Cross Validation*), ki poskrbi, da pri deljenju podatkov na k delov vsak del ohrani razmerja med razredi.

4.7 Optimizacija hiperparametrov

Pri učnih algoritmih smo omenili, da imajo različne parametre, ki jih določi uporabnik in da ti parametri lahko močno vplivajo na končni rezultat. Optimalni parametri so lahko zelo različni od problema do problema. Ravno zato je težko točno predvideti, katere vrednosti parametrov najbolje ustrezajo za izbrani model. Zato smo pred učenjem modelov izvedli še optimizacijo hiperparametrov. V splošnem optimizacija oz. iskanje hiperparametrov poteka tako:

1. Izberemo vrednosti za parametre, ki jih želimo optimizirati.
2. Podatke razdelimo na k enakih delov.
3. Model z izbranimi parametri naučimo na $k-1$ delih podatkov.
4. Model ocenimo na preostalem delu podatkov.

5. Koraka 3 in 4 ponovimo, tako da je vsak del podatkov enkrat testna množica (k-kratno prečno preverjanje).

6. Korake od 1 do 5 ponovimo za vse možne kombinacije vrednosti parametrov.

7. Izberemo parametre, ki so imeli v prečnem preverjanju najboljše povprečje ocen.

4.7.1 Izčrpno iskanje parametrov

Malheiro [46] omeni, da so izvedli optimizacijo hiperparametrov z izčrpnim preiskovanjem. Za vsak parameter, ki ga želimo optimizirati, podamo končno množico vrednosti. Algoritem preizkusi vse možne kombinacije vrednosti vseh parametrov in vrne tisto, ki se pokaže za najboljšo. Preizkusi jih s k-kratnim prečnim preverjanjem. Takšno preverjanje je lahko uporabno, če teh parametrov ni veliko in/ali so množice vrednosti majhne. Pri takšnem iskanju moramo točno določiti, katere vrednosti želimo preizkusiti. Tako za parameter, ki je definiran na intervalu $(0, \infty)$, lahko preizkusimo različne potence števila 10, npr. od 10^{-i} do 10^{i}.

4.7.2 Naključno iskanje parametrov

Pri večjem naboru parametrov se izkaže, da je zadovoljivo in manj zahtevno naključno iskanje parametrov [5]. Za razliko od izčrpnega iskanja, kjer podamo za vsak parameter končno množico vrednosti, tukaj podamo za vsak parameter distribucijo, iz katere se nato naključno vzorčijo vrednosti. Pri izčrpnem preiskovanju se iskalni prostor sistematično pregleda, vendar so točke iskanja fiksne. Če v teh fiksnih točah ni optimalne točke, potem izčrpno preiskovanje sploh nima možnosti najti optimalnih parametrov. Pri naključnem preiskovanju tega problema ni zaradi naključnega vzorčenja. Druga prednost, v primerjavi z izčrpnim iskanjem, se pokaže, ko imamo veliko število parametrov oz. velike zaloge vrednosti. Pri izčrpnem iskanju

je treba preiskati vse možne kombinacije, medtem ko lahko pri naključnem iskanju sami določimo, koliko iteracij želimo opraviti. Seveda, več iteracij, kot izvedemo, večja je verjetnost, da najdemo optimalne parametre. Recimo, da imamo tri parametre in za vsakega po 10 vrednosti, ki jih želimo preizkusiti. To je $10 * 10 * 10 = 1000$ kombinacij, ki jih mora izčrpni algoritem preizkusiti. Po drugi strani pa ima naključno iskanje pri 60 iteracijah več kot 95% verjetnosti, da najde vrednosti znotraj 5% območja okoli optimuma v iskalnem prostoru. Zadnjo trditev lahko enostavno dokažemo. V vsakem poskusu naključno izberemo vse parametre, kar pomeni, da izberemo točko v iskalnem prostoru. Zamislimo si 5% podprostora okoli optimuma v tem iskalnem prostoru. Verjetnost, da naključno izberemo točko v tem podprostoru okoli optimuma, je 0.05. Verjetnost, da izberemo točko izven tega podprostora, je $1 - 0.05$. Pri n poskusih je verjetnost, da nikoli ne izberemo točke iz tega podprostora $(1 - 0.05)^n$. Verjetnost, da vsaj enkrat izberemo točko iz tega podprostora v n poskusih, je potem $1 - (1 - 0.05)^n$. Če želimo, da je verjetnost, da tako točko najdemo, vsaj 95%, moramo rešiti enačbo $1 - (1 - 0.05)^n \geq 0.95$. Rešitev je $n \geq 59$.

Seveda naključno preiskovanje ne zagotavlja optimalnega rezultata, ki bi ga lahko našli z zelo natančnim izčrpnim preiskovanjem. Vendar pa je verjetnost, da najdemo skoraj optimalne parametre, zelo visoka. Pri velikem številu parametrov je izčrpno preiskovanje zelo drago in je zato naključno iskanje lahko bolj pametna izbira.

Spodaj je primer kode v Pythonu, kjer z naključnim iskanjem optimiziramo parametre za SVM klasifikator. Za naključno iskanje parametrov je v primeru uporabljena implementacija iz knjižnice scikit-learn, ki se imenuje RandomizedSearchCV. V spremenljivki *parameters* definiramo območje iskanja za vse parametre, ki jih želimo optimizirati. Za parameter *kernel* smo podali seznam vrednosti, za parameter C pa eksponentno distribucijo. Parameter *cv* v RandomizedSearchCV konstruktorju definira, s kolikokratnim prečnim preverjanjem želimo naključno izbrane parametre oceniti, za mero uspešnosti pa smo si izbrali mero F1 (parameter *scoring*). Podan je še

argument *n_iters*, ki definira, koliko naključnih kombinacij naj se preizkusi pri iskanju. S klicem metode *fit* začnemo iskanje parametrov, na koncu pa najboljše parametre izpišemo na zaslon.

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.svm import SVC
from scipy.stats import expon

parameters = {'kernel':('linear', 'rbf'), 'C':expon(scale=10)}

clf = RandomizedSearchCV(SVC(), parameters, n_iter=10, cv=10,
    scoring="f1_macro")
clf.fit(X_param , y_param)

print(clf.best_params_)
```

Primer kode 4.1: Primer uporabe RandomizedSearchCV

4.8 Ocenjevanje

4.8.1 Mera F1

Za oceno uspešnosti klasifikacije smo uporabili mero F1. Mera F1 je harmonično povprečje natančnosti (angl. *precision, Pr*) in priklica (angl. *recall, Re*). Natančnost predstavlja delež med številom pravilno napovedanih pozitivnih vrednosti in številom vseh pozitivno napovedanih vrednosti (pravilno in nepravilno napovedanih). Priklic pa je delež med številom pravilno napovedanih pozitivnih vrednosti in številom vseh dejansko pozitivnih vrednosti (ne glede na napoved). Mero F1 izračunamo po formuli

$$F1 = 2 * \frac{Pr * Re}{Pr + Re}$$

Mero F1 lahko na tak način uporabimo samo za binarne klasifikatorje (klasificiranje v dva razreda, pozitivnega in negativnega). Vendar pa z mero F1 ocenjujemo tudi večrazredne klasifikatorje. Mera $F1_{macro}$ se izračuna tako, da se mera $F1$ izračuna za vsak razred posebej (instance razreda so

pozitivni primeri, vse ostalo pa negativni) in potem vse mere povprečimo. Formula za mero $F1_{macro}$, pri n razredih, je

$$F1_{macro} = \frac{1}{n}\sum_{i=0}^{n} 2 * \frac{Pr_i * Re_i}{Pr_i + Re_i}$$

4.8.2 Mera R2

Za oceno uspešnosti regresorjev smo uporabili determinacijski koeficient R^2. Ta mera oceni, kako dobro se model prilega podatkom glede na to, kolikšen del variance znanih vrednosti pojasnimo z napovedanimi vrednostmi. Vrednost 1 pomeni, da se model popolnoma prilega podatkom. Negativna vrednost pomeni, da se model podatkom sploh ne prilega. Denimo, da imamo znane vrednosti $y_0 ... y_n$ in njihove napovedane vrednosti $f_0 ... f_n$. $\overline{y}$ je povprečje znanih vrednosti. Determinacijski koeficient R^2 se izračuna po formuli

$$R^2 = 1 - \frac{\sum_{i=0}^{n}(y_i - f_i)^2}{\sum_{i=0}^{n}(y_i - \overline{y})^2}$$

Poglavje 5

Rešitev

V tem poglavju predstavimo, kako smo implementirali naš sistem. Za programiranje naše rešitve smo si izbrali programski jezik Python, ki je tudi drugače priljubljen jezik na področju strojnega učenja.

5.1 Pridobitev besedil

Prvi korak je bil pridobiti vsa besedila, na katerih bomo lahko učili in testirali naše modele. Malheiro [46] je javno objavil seznam anotiranih besedil, na podlagi katerih smo delali tudi mi. Zaradi avtorskih pravic dejanska besedila niso bila objavljena, vendar le spletni naslov, od koder so bila besedila prenesena. Seznam vseh spletnih naslovov je zapisan v datoteki tipa CSV (angl. *Comma Seperated Values*). Datoteke CSV imajo končnico `.csv`. Vsaka vrstica v `.csv` datoteki predstavlja eno besedilo (učni primer). Vrstica v datoteki za učno množico vsebuje naslov pesmi, ime izvajalca pesmi, ID besedila, podatke o valenci in aktivnosti čustev (povprečje in standardni odklon) in spletni naslov. Vrstica v datoteki za testno množico vsebuje ID besedila, kvadrant, v katerega spada besedilo, ime izvajalca pesmi, naslov pesmi in spletni naslov. Vrsticam v datoteki za učno množico smo zaradi praktičnosti naknadno dodali še podatek o kvadrantu, v katerega pesem spada, izračunan na podlagi podatkov o povprečju valence in aktivnosti. Nekateri izmed sple-

tnih naslovov niso bili dosegljivi, zato smo te naslove nadomestili z novimi spletnimi naslovi. Ugotovili smo, da je v testni in učni množici nekaj duplikatov. Nekatera besedila iz učne množice so se ponovila v testni množici. V testni množici so bila besedila, ki so se tudi po trikrat ponovila, v enem primeru so bili celo razredi med njimi različni. Tako smo vsega skupaj odstranili 9 (podvojenih) besedil iz testne množice. Tako je na koncu učno množico sestavljalo 180 besedil in testno 762 besedil.

Za odpiranje in shranjevanje datotek tipa CSV smo uporabili knjižnico Pandas[1]. Funkcija `pandas.read_csv` prebere vsebino CSV datoteke v objekt `pandas.DataFrame`. `pandas.DataFrame` tudi drugače uporabljamo čez celoten sistem za prenos podatkov, kot so značilke.

Nato smo s pomočjo knjižnice urllib[2] prenesli HTML (angl. *Hyper Text Markup Language*) vsebino iz spletnega naslova. S knjižnico Beautiful Soup 4[3] smo potem iz prenesene HTML vsebine izluščili glasbeno besedilo. Besedilo smo potem lokalno shranili v tekstovno datoteko. To smo ponovili za vse spletne naslove.

Na spletnih straneh iz različnih virov je besedilo drugje v HTML strukturi, zato je bilo treba prej ročno preveriti vsak spletni vir. Na primer, na straneh na naslovu `www.metrolyrics.com` je besedilo v HTML znački `<div class="lyrics-body>` in vsi verzi so znotraj HTML značk `<p class="verse>`. Naš sistem trenutno zna izluščiti glasbeno besedilo s spletnih strani iz 24 različnih virov. Če je potreba po še dodanih virih, pa se da sistem hitro nadgraditi.

Spodaj sledi primer programske kode za prenos glasbenih besedil s spleta. Na takšen način smo tudi mi prenesli glasbena besedila s spleta, vendar pa je spodnji primer vseeno zelo poenostavljen.

```
1 # Naložimo Python module Pandas, urllib in BeautifulSoup
2 import pandas as pd
3 from urllib.request import urlopen
```

[1] `http://pandas.pydata.org/`
[2] `https://docs.python.org/3/library/urllib.html`
[3] `https://www.crummy.com/software/BeautifulSoup/`

```
from bs4 import BeautifulSoup

# Preberemo CSV datoteko
dataset = pd.read_csv('dataset.csv')

# Zanka čez vse vrstice
for index,row in dataset.iterrows():
    # Iz spletnega naslova prenesmo HTML vsebino
    source = row['Source']
    html_data = urlopen(source).read()

    # Iz HTML vsebine naredimo 'juho'
    soup = BeautifulSoup(html_data, 'html')

    # Iz 'juhe' preberemo vse verze in jih združimo v besedilo
    verses = soup.find_all('p', 'verse')
    verses = [verse.get_text() for verse in verses]
    lyrics = '\n\n'.join(verses)

    # Besedilo shranimo v tekstovno datoteko
    song_id = row['Id']
    with open(song_id+'.txt', 'w+') as output_file:
        output_file.write(lyrics)
```

Primer kode 5.1: Primer pridobivanja glasbenih besedil iz spleta

5.2 Preprocesiranje besedil

Besedila, ki smo jih prenesli s spleta, so največkrat vsebovala vrsto artefaktov. Pogosti artefakti so bile značke (npr. za oznako refrena), podatki o pesmi (naslov, izvajalec) ali zapisani odmevi in zvoki v avdio ozadju. Veliko takih artefaktov lahko pokvari analizo besedil, predvsem pri analizi strukture besedil. Zato smo iz besedil odstranili takšne artefakte. Poleg čiščenja besedil smo za vsako besedilo določili še refren. Besedila smo tudi še označili s POS oznakami in jih lematizirali.

5.2.1 Podatki o besedilu

Najprej smo iz besedil odstranili podatke o besedilu, kot so naslov pesmi, ime izvajalca, naslov albuma in leto izida. V vseh besedilih, ki smo jih pregledali, so se ti podatki pojavili ali v prvih ali v zadnjih nekaj vrsticah besedila. Pogosta kombinacija je, da je v prvi vrstici ime izvajalca pesmi, v drugi vrstici naslov albuma in v tretji vrstici naslov pesmi. Algoritem za brisanje teh podatkov pogleda vrstice prvega in zadnjega odstavka (deli besedila ločeni z „`\n\n`“ in primerja te vrstice z imenom izvajalca pesmi in naslovom pesmi. To sta namreč edina dva podatka, ki jih vemo o besedilu. Algoritem ne deluje na čisti podobnosti, vendar pa s pomočjo knjižnice Fuzzywuzzy[4] primerja vrstice glede na razmerje podobnosti, ki mora biti večje od določenega praga. Ta prag je bil ocenjen na podlagi poskusov. Algoritem tudi odstrani vrstice, ki vsebujejo nekatere besedne zveze, kot so „text by“ ali pa „lyrics by“.

5.2.2 Prevod besedila

Nekatera besedila v originalu niso v angleščini in sta bila zato zapisana original in prevod besedila. Nas so zanimala samo angleška besedila in smo zato v takem primeru odstranili originalno (ne-angleško) besedilo. Teh besedil je bilo zelo malo in angleški prevodi so bili zelo specifično označeni. Vedno je bilo originalno besedilo na začetku in nato za njim angleški prevod. Angleški prevod se je dalo prepoznati po tem, da sledi vrstici, ki vsebuje „English:“ ali „English translation:“. Dvopičje je tukaj zelo pomembno. Tako je naš algoritem za ekstrakcijo angleških prevodov zelo enostaven, preprosto odstrani vse vrstice do vključno vrstice, ki vsebuje eno od prej omenjenih nizov. Algoritem je preprost in zelo specifičen za besedila, s katerimi smo delali, vendar je svoje delo opravil.

Če bi želeli algoritem narediti bolj splošen, bi morali upoštevati več možnih scenarijev. Prevod bi se lahko pojavil pred originalom ali pa bi bilo originalno besedilo in prevodi pomešani. Lahko bi zgradili algoritem, ki

[4] `https://github.com/seatgeek/fuzzywuzzy`

bi preverjal posamezne kitice ali pa vrstice in računal delež angleških besed, čeprav je pri zelo majhnem številu besed to lahko nezanesljivo. Na podlagi tega bi se potem lahko odločili, katere kitice ali vrstice obdržati. Za Python obstaja več knjižnic za prepoznavanje jezika iz besedila, dve sta langdetect[5] in polyglot[6].

5.2.3 Značke

Značke lahko največkrat prepoznamo po oglatih oklepajih. Pogosto nad refrenom vidimo značko `[Chorus]`. Tovrstne značke so nam koristile, ko smo iskali refren v besedilu. Koristile so nam tudi še značke za ponovitev npr. `[x3]`, `[2 times]`. Takšne značke smo poskušali interpretirati in ponoviti del besedila, na katerega se navezujejo.

Je pa bilo v veliko besedilih še mnogo drugih značk, kot so `[fade]`, `[spoken]`, `[Barbara:]` ipd. Takšne značke za nas niso imele nobenega pomena in smo jih zato preprosto odstranili. Iskanje in brisanje značk smo izvedli s pomočjo regularnih izrazov (angl. *Regular Expression, Regex*). Standardna knjižnica v Pythonu za regularne izraze se imenuje re[7].

5.2.4 Iskanje refrena

Že vnaprej smo vedeli, da bomo morali za vsako besedilo poznati refren (če sploh obstaja), saj smo pozneje na podlagi refrena izračunali nekaj značilk. Najbolj preprosto je bilo v besedilih prepoznati refren, če je bil že označen z značko za refren (poleg besede „Chorus“ se za refren uporabljata še „Hook“ predvsem pri rap besedilih in pa „Refrain“). V primeru, ko teh značk ni bilo prisotnih v besedilu, smo poskušali na podlagi določenih predpostavk poiskati refren. Najpomembnejša predpostavka je to, da je refren tista kitica, ki se bolj ali manj enaka ponovi največkrat v besedilu. To je ne nazadnje tudi definicija refrena (*besedilo, ki se v pesmi redno ponavlja* [56]). Dodatne

[5] `https://pypi.python.org/pypi/langdetect`
[6] `https://pypi.python.org/pypi/polyglot`
[7] `https://docs.python.org/3/library/re.html`

predpostavke so še, da je refren najverjetneje že ena izmed kitic na začetku besedila in pa da je kitica, v kateri se pojavi naslov, še bolj verjeten kandidat za refren. Algoritem tako gre čez vse kitice in za vsako glasuje glede na hevristike. Kitica, ki na koncu zbere največ glasov, se predpostavi, da je refren. Če imata dve kitici enako število točk, potem algoritem refrena ni našel in predpostavimo, da ne obstaja.

5.2.5 Čiščenje

Zadnji korak pri čiščenju besedil je bil še preprosto odstraniti vse neželene in odvečne znake. Do tega koraka smo značke, ki smo jih potrebovali, že uporabili in nam na tej točki ne koristijo več. S pomočjo regularnih izrazov smo odstranili vse dele besedila, ki so znotraj oglatih oklepajev. Besedilo, ki je znotraj oklepajev in zavitih oklepajev, je navadno odmev iz ozadja ali vzklic. Takšna besedila smo obdržali, odstranili smo zgolj oklepaje. Odstranili smo tudi vse odvečne presledke in znake za novo vrstico. Na primer, kjer je več presledkov zaporedoma, smo jih skrajšali v enega, pri znakih za novo vrstico pa smo tri ali več znakov skrajšali v dva. Tako lahko kasneje predpostavljamo, da so vse kitice med seboj ločene z „`\n\n`“.

5.2.6 POS označevanje

Očiščena besedila smo nato še označili s POS oznakami. Poleg finih in grobih POS oznak smo besedam določili še leme in korene. Rezultate smo predstavili kot tabele. Vrstica v tabeli predstavlja po eno besedo oz. žeton, po stolpcih pa so POS oznake, leme in koreni. Te tabele smo nato shranili kot CSV datoteke.

5.2.7 Primer

Spodaj je primer glasbenega besedila z različnimi artefakti: v prvih treh vrsticah so podatki o pesmi in avtorju, vse kitice so označene z značkami, v

oklepajih in na sredini besedila so deli, ki se navezujejo na glasbeno podlago in pa v besedilu sta tudi še dve navodili za ponovitev vrstice ali kitice.

```
A Little Kiss
Phrance Presheren
1922

[Verse 1]
When I see you
On the corner
My heart just skips a beat (beat)

[Chorus]
All it took was a little kiss x2
Never thought I'd feel like this

[Verse 2]
When you're near me
In the hall ways
I'm trying not to stare

[Instrumental]

[Chorus] x2
```

Po čiščenju je besedilo videti tako:

```
When I see you
On the corner
My heart just skips a beat beat

All it took was a little kiss
All it took was a little kiss
Never thought I'd feel like this

When you're near me
In the hall ways
I'm trying not to stare
```

```
All it took was a little kiss
All it took was a little kiss
Never thought I'd feel like this

All it took was a little kiss
All it took was a little kiss
Never thought I'd feel like this
```

Še primer, kako je videti lematizirana druga kitica:

```
when you be near i
in the hall way
i be try not to stare
```

5.3 Priprava značilk

Ko so bili dokumenti pripravljeni, smo iz njih pridobili značilke. Značilke, ki smo jih pridobili, so raznovrstne, zato smo uporabili različne načine pridobivanja značilk.

Vsebinske značilke. Vsebinske značilke smo vse pridobili s scikit-learn razredom TfidfVectorizer. TfidfVectorizer lahko uporabimo za računanje pogostosti simbolov (angl. *Term Frequency, tf*) ali mere „pogostost simbolov - inverzna pogostost v dokumentih“ (angl. *Term Frequency – Inverse Document Frequency, tf-idf*). tf-idf je mera, ki uteži pogostosti simbolov v dokumentu glede na to, kako pomembna je katera beseda v celotnem korpusu. Mera tf-idf se pogosto uporablja v tekstovnem rudarjenju. TfidfVectorizer-ju lahko podamo argument *ngram_range*, da določimo, kakšen model naj zgradi (unigram, bigram itd.). TfidfVectorizer lahko tudi normalizira vektorje izrazov, pri čemer smo mi uporabili L2 normalizacijo.

Spodaj je preprost primer, kako uporabiti TfidfVectorizer za računanje pogostosti simbolov ali tf-idf mer. TfidfVectorizer sprejeme vrsto parametrov. *use_idf* določa, ali naj računa pogostost simbolov ali tf-idf. *norm* določa, ali naj rezultat normalizira in na kakšen način (L1 ali L2). Če po-

damo parametra *use_idf=False* in *norm=None*, potem se vede isto kot CountVectorizer.

```
from sklearn.feature_extraction.text import TfidfVectorizer

train_set = ("The sky is blue.", "The sun is bright.")
test_set = ("The sun in the sky is bright.", "We can see the shining sun,
    the bright sun.")

vectorizer = TfidfVectorizer(analyzer='word', ngram_range=(1,1),
    use_idf=False, norm=None)
train_features = vectorizer.fit_transform(train_set).todense()

print(vectorizer.vocabulary_)
# {'blue': 0, 'bright': 1, 'is': 2, 'sky': 3, 'sun': 4, 'the': 5}

test_features = vectorizer.transform(test_set).todense()
print(test_features)
# [[ 0.  1.  1.  1.  1.  2.]
#  [ 0.  1.  0.  0.  2.  2.]]
```

Primer kode 5.2: Primer uporabe TfidfVectorizer

Stilistične značilke. Pri stilističnih značilkah smo zastavili dva modela. Prvi temelji na štetju besednih vrst (POS oznak) in ravno tako uporablja TfidfVectorizer. TfidfVectorizerju lahko podamo svoje besedišče (parameter *vocabulary*), da potem upošteva samo podane besede. Zgradili smo dva modela, enega za računanje pogostosti simbolov in enega za tf-idf.

```
my_POS_vocab = ('VRB', 'NN', 'ADJ', 'PRN', 'NUM')
pos_vectorizer = TfidfVectorizer(analyzer='word',
    vocabulary=my_POS_vocab, use_idf=False, norm='l2')
```

Primer kode 5.3: Primer uporabe TfidfVectorizer kjer podamo seznam besed

Drugi model je sestavljen iz štetja slengovskih izrazov v besedilu in štetja besed z velikimi začetnicami in velikimi črkami. Za štetje slengovskih besed smo uporabili CountVectorizer, ki smo mu podali naš slovar slengovskih besed. Upoštevali smo samo skupno število vseh slengizmov v dokumentu, zato smo vse pogostosti simbolov sešteli. Za štetje besed z velikimi črkami pa smo uporabili preprosto zanko, ki gre čez vse besede in poveča števec, če je beseda v celoti zapisana z velikimi črkami oz. ima veliko začetnico (dva različna števca). Upoštevali smo samo tiste besede z veliko začetnico, ki niso prva beseda v vrstici.

Značilke strukture pesmi. Pri značilkah strukture pesmi smo šteli stvari, kot so število ponovitev naslova v besedilu ali pa število vseh kitic. Deloma smo to implementirali z regularnimi izrazi, deloma s števci in zanko. Ker smo že prej poskrbeli, da smo v besedilu prepoznali refren, je bilo računanje značilk strukture pesmi mnogo lažje. Besedilo smo ločili v kitice tako, da smo ga „razrezali“, kjer se pojavi niz „`\n\n`“. Potem smo posamezne kitice lahko označili kot refren, če so bile dovolj podobne besedilu, ki smo ga predhodno prepoznali kot refren.

Semantične značilke. Za računanje semantičnih značilk smo uporabili vrsto že obstoječih orodij. Eno izmed njih je Synesketch, ki pa je implementiran v Javi, tako da ga neposredno nismo mogli uporabiti v naši Pythonski kodi. Vse, kar smo želeli, je, da besedilo pošljemo Javanskemu objektu iz Synesketcha in od njega nazaj dobimo seznam vrednosti. Zato smo upora-

bili knjižnico Py4J[8], ki omogoča, da Python program dostopa do Javanskih objektov v Java virtualnem stroju (angl. *Java Virtual Machine*). Py4J deluje tako, da napišemo program v Javi, ki zažene `py4j.GatewayServer` in izpostavi objekte, ki bi jih radi dostopali iz Python programa. Nato se Python program poveže na ta strežnik, od koder lahko dobi želene Javanske objekte. Ti objekti se v Python kodi uporabljajo tako, kot da so Python objekti. Na tak način smo pridobili čustvene uteži, ki jih iz besedila zna izračunati Synesketch.

Ostala orodja, ki smo jih uporabili, so semantični slovarji DAL, ANEW, Warrinerjev slovar [68] in GI. GI značilke smo iz besedil pridobili tako, da smo najprej pripravili CountVectorizer in mu za besedišče dali seznam vseh GI-jevih kategorij. Potem smo v zanki šli čez vse besede v besedilu in besede, ki so v GI slovarju zamenjali s seznamom kategorij, v katere spadajo, ostale besede pa smo odstranili. Nato smo tako transformirane dokumente s CountVectorizerjem pretvorili v matriko pogostosti simbolov. To pomeni, da smo za vsak dokument dobili podatek o tem, koliko besed spada v katero kategorijo.

Warrinerjev slovar [68] in slovarja DAL in ANEW smo uporabili na enak način. V besedilu smo prešteli vse pojavitve besed, ki so v slovarjih, in nato izračunali povprečne vrednosti besed za čustvene dimenzije, ki jih slovarji vsebujejo.

Malheiro [46] je slovarja DAL in ANEW uporabil še na en način, iz njih je izpeljal slovarje besede za vsak kvadrant posebej. Slovarje za kvadrante je v grobem zgradil tako:

1. Najprej je za vsak kvadrant izbral 18 izhodiščnih besed. To so besede (za čustva), ki so definirane v Russellovem krožnem modelu2.1.

2. Nato je iz izhodiščnih besed odstranil besede, ki niso v DAL in ANEW slovarju.

[8] `https://www.py4j.org/`

3. Besedam, ki so ostale, je poiskal čustvene sopomenke s pomočjo Wordnet Affect-a[9]. V slovarje je dodal samo besede, ki so v DAL ali ANEW.

4. Slovarje je nato razširili še s sopomenkami trenutnih besed v slovarju. Za iskanje sopomenk je uporabili Wordnet[10]. Dodali so samo besede, ki so v DAL ali ANEW.

Tako je zgradil za vsak kvadrant svoj slovar besed skupaj s podatkoma o valenci in aktivnosti za vsako besedo. Preden je bila katera koli beseda vstavljena v slovar, sta njeno čustveno vrednost potrdili še dve osebi. Obe osebi sta se morali strinjati, da je bila beseda nato dejansko vstavljena v slovar. DAL in ANEW uporabljata različni skali, DAL od 1 do 3, ANEW pa od 1 do 9, zato je vrednosti iz ANEW skaliral na DAL-ovo lestvico.

Po istem postopku smo tudi mi poskušali zgraditi slovarje za kvadrante, vendar pa so se naši rezultati močno razlikovali od njegovih. Število besed, ki smo jih mi na koncu postopka imeli v vsakem kvadrantu, je bilo nekajkrat manjše. Na primer, v slovarju za kvadrant Q1 je Melheiro [46] imel 132 besed, medtem ko smo jih mi dobili samo 25. Zakaj se točno tak postopek pri nas ni obnesel, ne moremo pojasniti. En razlog bi lahko bil, da morda v navodila niso vključeni vsi detajli, vendar pa se zdi postopek vseeno dobro opisan. Kljub temu smo poskušali in z različnimi spremembami v korakih doseči podobne rezultate (vsaj približno enako število besed v slovarjih). Poskušali smo, če besede sproti ne odstranjujemo iz slovarja, če niso v DAl ali ANEW, ampak to storimo čisto na koncu. Ali pa da 4. korak (iskanje sinonimov) ponovimo večkrat, tako da dobimo sinonime sinonimov. Nazadnje smo zgradili naše slovarje tako, da smo začeli z razširjenim seznamom začetnih besed. To pomeni, da smo, namesto da bi imeli v začetnih besedah samo besedo „excited“, dodali še besedi „excitement“ in „exciting“. Potem smo slovarje besed po vsakem koraku razširili še z njihovimi lemami. Iskanje sinonimov smo izvedli dvakrat, tako da smo dobili sinonime sinonimov

[9] `http://wndomains.fbk.eu/wnaffect.html`
[10] `http://www.nltk.org/howto/wordnet.html`

besed, ki so bile v slovarju. Besed, ki niso v DAL in ANEW, nismo odstranjevali iz slovarjev po vsakem koraku, ampak šele čisto na koncu postopka. V slovarju smo na koncu obdržali samo besede, ki so glede na valenco in aktivnost nedvoumno v enem kvadrantu, to pomeni, da ne ležijo na meji med dvema ali več kvadranti. S takim postopkom smo dobili primerljivo število besed v slovarjih. Tudi pozneje klasifikacijski rezultati na podlagi značilk iz teh slovarjev dosegajo podobne rezultate, kot so jih v Malheirovi študiji [46]. Vendar pa se nam je tak postopek že od začetka zdel preveč kompliciran in smo ga poenostavili. Preprosto smo iz DAL in ANEW vzeli vse besede, ki nedvoumno spadajo v en kvadrant in iz teh besed sestavili slovarje po kvadrantih. Model iz teh značilk dosega boljše rezultate kot pa model, zgrajen po prej omenjenem postopku.

5.3.1 Primer

Za besedilo v poglavju 5.2.7 smo izračunali značilke iz nekaterih modelov. V tem poglavju predstavimo nekatere od teh izračunanih značilk.

V tabeli 5.1 so izračunane nekatere stilistične značilke. Značilka *#Slang* pomeni število slengovskih besed v besedilu. Značilka *ACL* (angl. *All Capital Letters*) pomeni število vseh besed, ki so zapisane z velikimi črkami, *FCL* (angl. *First Capital Letter*) pa število vseh besed, ki se začnejo z veliko začetnico (ne upoštevamo prvih besed v vrsticah). Program je prepoznal 26 slengovskih besed, čeprav besedilo dejansko ne vsebuje slengovskih besed. Za to obstajata dva razloga. Prvi je ta, da slovar slengovskih besed vsebuje tudi „običajne“ besede, ki pa imajo poleg primarnega pomena še slengovski pomen. Tako na primer beseda „beat“, ki se v pesmi navezuje na bitje srca, lahko v slengu tudi pomeni „dolgočasen“ [3]. Drugi razlog je ta, da smo uporabili slovar slengovskih besed z vsemi variacijami besed in besednih zvez, tako da se posebej štejejo „beat“ in „a beat“, kar dodatno poveča končno številko.

V tabeli 5.2 so nekatere od izračunanih značilk strukture pesmi. Značilke *#CH*, *#Title*, *#VorC* in *#V* predstavljajo število ponovitev refrena, naslova,

#Slang	ACL	FCL
26	1	3

Tabela 5.1: Primer stilističnih značilk.

#CH	#Title	#VorC	#V	C...	>2CAtTheEnd	VCCVCC...
3.0	6.0	5.0	2.0	False	True	True

Tabela 5.2: Primer nekaterih značilk strukture pesmi.

število vseh kitic in število kitic, ki niso refren. Značilka *C...* pomeni, ali se besedilo začne z refrenom, *>2CAtTheEnd* pomeni, ali se besedilo konča vsaj z dvema ponovitvama refrena in *VCCVCC ...* pomeni, ali je med vsemi kiticami, ki niso refren, vsaj en refren.

V tabeli 5.3 so primeri nekaterih semantičnih značilk na podlagi slovarjev DAL in ANEW ter slovarjev, ki smo jih zgradili za vsak kvadrant. Značilke tipa *#GAZQ* pomenijo število besed, ki so v slovarjih za kvadrante Q1, Q2, Q3 ali Q4. Značilki „VinGAZQ1Q2Q3Q4“ in „AinGAZQ1Q2Q3Q4“ pomenita povprečno valenco in aktivnost besed, ki so v slovarjih za kvadrante. Značilke „VinDAL“, „AinDAL“, „VinANEW“ in „AinANEW“ pomenijo povprečne vrednosti za valenco in aktivnost besed, ki so v slovarjih DAL in ANEW.

V tabeli 5.4 so nekatere semantične značilke, izračunane na podlagi slovarja GI. Vsaka značilka predstavlja število besed, ki spadajo v to določeno kategorijo. Tako npr. značilka „BodyPt“ pomeni število besed v besedilu, ki spadajo v kategorijo „deli telesa“.

#GAZQ1	#GAZQ2	#GAZQ3	#GAZQ4	VinGAZQ1Q2Q3Q4
4	2	1	0	2.025

AinGAZQ1Q2Q3Q4	VinDAL	AinDAL	VinANEW	AinANEW
2.135	1.914	1.706	2.579	2.310

Tabela 5.3: Primer nekaterih značilk na podlagi DAL in ANEW slovarjev.

Active	BodyPt	Negate	Passive	Positiv	Self	Space	Weak	You
13	1	4	4	6	7	4	6	2

Tabela 5.4: Primer nekaterih GI značilk.

V tabeli 5.5 so značilke, ki jih je izračunal Synesketch. Prvih šest značilk predstavlja uteži za posamezna čustva, značilka „General“ predstavlja vsa čustva skupaj in značilka „Valence“ predstavlja valenco besedila. Uteži imajo vrednosti med 0 in 1, kjer 0 pomeni, da čustvo ni izraženo in 1 pomeni, da je čustvo popolnoma izraženo. Valenca ima lahko vrednosti -1, 0 in 1 (negativna, nevtralna in pozitivna čustva).

Happiness	Sadness	Anger	Fear	Disgust	Surprise
0.267	0.267	0.133	0.133	0.133	0.032

General	Valence
0.647	1.0

Tabela 5.5: Primer Synesketch značilk.

5.4 Učenje modelov

Postopek učenja je potekal tako, da smo si najprej izračunali vse značilke za učno in testno množico. Nato smo naslednje značilke še skalirali s scikit-learn StandardScaler-jem: značilke v povezavi s številom slengovskih besed in besed z velikimi začetnicami, vse značilke strukture pesmi, značilke iz Synesketch-a in značilke na podlagi DAL, ANEW in Warriner [68] slovarjev. Z modulom joblib[11] smo nato vse značilke shranili kot stisnjen serializiran Python objekt („pickle“). Tako smo se izognili vnovičnemu računanju značilk.

Nato smo značilke po modelih z algoritmom ReliefF (in RFE za regresijo) razvrstili po pomembnosti in izbrali n najboljših. Včasih se da pri ocenah, ki

[11] `https://pythonhosted.org/joblib/`

jih poda ReliefF, zelo očitno razbrati, katere značilke so zelo dobre in katere ne, saj je v ocenah velik razkorak. Vendar pa to ne drži vedno in lahko ocene zelo enakomerno padajo, zato smo n določili tudi s poskušanjem različnih vrednosti npr. od 100 do 1000 s korakom 100. Malheiro [46] je objavil število izbranih značilk za posamezne modele, tako da je tudi to vplivalo pri izbiri n-ja. Z izbiro najboljših značilk smo poskušali zmanjšati število vseh značilk za učenje. Manjše število značilk lahki prinese dve prednosti. Učenje na manjšem številu značilk vzame manj časa. Veliko število značilk pri majhnem številu učnih primerov lahko vodi v pretirano prilagajanje. Po drugi strani tudi izbira značilk vodi v isti problem, saj smo izbrali take značilke, ki se dobro prilagajajo učnim primerom.

Na izbranih značilkah smo se potem z učnimi algoritmi SVM in XGBoost naučili različne klasifikatorje in regresorje. Parametre za učne algoritme smo izbrali z naključnim preiskovanjem. Pri naključnem iskanju parametrov je treba za vsak parameter, ki ga želimo optimizirati, podati distribucijo vrednosti, iz katere se potem vlečejo naključne vrednosti. Za implementacijo algoritma SVM za klasifikacijo, ki se v scikit-learn imenuje SVC, smo podali distribucije za naslednje parametre: *C*, *degree*, *gamma*, *coef0* in *kernel*. Pri implementaciji algoritma SVM za regresijo (SVR) smo optimizirali še dodaten parameter *epsilon*. Pri parametru *kernel*, tj. vrsta jedra, ki naj ga SVM uporabi, smo izbirali med linearnim, polinomskim in RBF jedrom. Pri parametru *degree*, ki določi stopnjo polinoma in se upošteva zgolj pri polinomskem jedru, smo izbirali med vrednostmi od 1 do 5. Za ostale parametre smo določili uniformne distribucije na intervalih od 0 in 10^n. n smo izbirali naključno na intervalu med -4 in 4. To pomeni, da smo za te parametre podali naključne uniformne distribucije na intervalih med 0 in 10^n, kjer je $-4 < n < 4$. Tako smo za te parametre preiskali manjše vrednosti (med 0 in 0.0001) in tudi večje (do 10.000). Za algoritem XGBoost smo optimizirali naslednje parametre: *colsample_bytree*, *gamma*, *learning_rate*, *max_depth*, *min_child_weight*, *n_estimators*, *reg_alpha*, *subsample*. Pri optimizaciji teh parametrov skale distribucij nismo naključno izbirali, ampak smo jih že vnaprej

določili.

Optimizacija parametrov je bila nazadnje videti tako, da smo nekajkrat izbrali naključne skale distribucij in potem iz teh distribucij 60-krat naključno vzorčili vrednosti parametrov, klasifikatorje s temi parametri pa ocenili s 4-kratnim prečnim preverjanjem. Najboljši model vsake iteracije smo nato še ocenili z 10-kratnim prečnim preverjanjem s tremi ponovitvami. Za izbiro najboljšega modela smo različne modele primerjali s to oceno.

5.5 Ocenjevanje modelov

Izbrane modele smo dokončno ocenili z 10-kratnim prečnim preverjanjem z desetimi ponovitvami na učnih podatkih. Končna ocena je povprečje vseh rezultatov. Ocene, pridobljene na tak način, poroča tudi Malheiro [46], zato smo modele tako ocenili tudi mi. Računali smo še eno dodatno oceno, in sicer tako, da smo posamičen model naučili na vseh učnih primerih in nato model ocenili na testni množici. Takšna ocena je veliko manj optimistična in je boljša mera za kakovost modela. V naslednjem poglavju te ocene za najboljše modele tudi predstavimo.

Poglavje 6

Rezultati

V tem poglavju predstavimo rezultate dokončno izbranih modelov. Opravili smo veliko poskusov učenja posameznih in kombiniranih modelov. Tukaj predstavimo ocene uspešnosti nekaterih modelov. Oznake v imenih modelov pomenijo naslednje: „st“ pomeni, da smo besedila lematizirali, „sw“ pomeni, da smo iz besedil odstranili prepovedane besede, „pos“ pomeni, da smo uporabili POS značke in tf-idf pomeni, da smo uporabili mero tf-idf in ne pogostost simbolov. Modeli, katerih ime se konča z „- XGB“, so bili naučeni z algoritmom XGBoost.

Tabele z rezultati so zgrajene tako: stolpec „Model“ vsebuje ime modela, stolpec „#značilk“ pove število izbranih značilk za ta model, stolpec „10xCV10“ podaja povprečni rezultat 10-kratnega prečnega preverjanja (10 ponovitev) na učni množici, stolpec „Malh.“ podaja oceno iz Malheriove študije [46] (če ta obstaja) za primerjavo z oceno v stolpcu „10xCV10“ in stolpec „u-t“ pove oceno, ki smo jo dobili tako, da smo model naučili na učnih množici (180 besedil) in izračunali oceno na testni množici (762 besedil). Modeli so pri klasifikaciji ocenjeni z mero F1, pri regresiji pa z mero R^2.

6.1 Regresija

Pri regresiji smo poskušali besedilom napovedati vrednosti za valenco in aktivnost na intervalu od 1.0 do 3.0. V tabeli 6.1 so rezultati za regresijo glede na valenco, v tabeli 6.2 pa so rezultati regresije glede na aktivnost. Malheiro [46] poroča rezultate samo za dva končna kombinirana modela. Model za regresijo glede na aktivnost z 234 značilkami je pri njih dosegel R^2 rezultat 0.59. Model za regresijo glede na valenco s 340 značilkami je pri njih dosegel R^2 rezultat 0.61. S podobnimi modeli smo dosegli boljše rezultate, in sicer za valenco 0.81 ter za aktivnost 0.72. Potrdili smo tudi, da je napovedovanje vrednosti valence natančnejše kot pa aktivnosti. Vendar pa smo glede teh rezultatov dokaj skeptični, saj je ocena podana na učnih podatkih (ker testna množica nima podatkov o valenci in aktivnosti). Za to smo poskusili, kakšno F1 oceno dobimo, če na podlagi regresorjev klasificiramo testna besedila v poloble (pozitivna-negativna valenca, nizka-visoka aktivnost). Za klasificiranje na podlagi regresije smo za valenco dosegli oceno 55% in za aktivnost oceno 54%. Modeli, naučeni z algoritmom XGBoost, za regresijo so dosegli slabše rezultate kot ostali najboljši modeli, naučeni s SVM.

6.2 Klasifikacija

6.2.1 Kvadranti

Največ poskusov je bilo opravljenih ravno pri klasifikaciji v kvadrante. Pri tem problemu smo tudi dosegali najslabše rezultate. V tabeli 6.3 so predstavljeni nekateri od teh rezultatov. Predvsem smo razočarani nad našimi vsebinskimi značilkami, ki so se veliko slabše obnesle kot pri Malheiru [46]. Smo pa zadovoljni s tem, kako smo pripravili nove predlagane značilke (značilke strukture, stila in semantične značilke), ki se obnesejo primerljivo s tistimi v izvorni študiji. V kombiniranih modelih se verjetno kaže tudi odsotnost modela LIWC značilk, ki je pri Malheiru [46] dosegel F1 oceno 71.1%. Model, naučen z XGBoost, je dosegel podoben rezultat kot najboljši SVM model.

Model	#značilk	10xCV10
(M1X4T) unigrami+st - tfidf	400	0.58
(M1X3T) unigrami - tfidf	400	0.56
(M21) #POS_značke	17	0.53
(M23) #POS_značke - tfidf	10	0.81
(M42) DAL+ANEW	13	0.39
(M42X) DAL+ANEW - celotna	18	0.44
(M43) GI	55	0.50
(M43X) GI - tfidf	72	0.55
(M44) Synesketch	4	0.86
(M45) Warriner	20	0.50
C1RV (M1X3T, M1X4T)	450	0.66
C4RV (M44)	10	0.13
C124RV (C1RV, M23, C4RV)	450	0.24
vsi modeli	300	0.81
vsi modeli - XGB	200	0.65

Tabela 6.1: Ocene modelov pri regresiji glede na valenco.

Model	#značilk	10xCV10
(M12T) trigrami+pos - tidf	500	0.46
(M1X3T) unigrami - tfidf	300	0.43
(M1X4T) unigrami+st - tfidf	200	0.42
(M1X1T) 4grami+pos - tfidf	900	0.42
(M43X) GI - tfidf	55	0.42
C1AV (M12T, M1X1T, M1X3T, M1X4T)	600	0.64
C14AV (C14AV, M43x)	300	0.70
vsi modeli	300	0.72
vsi modeli - XGB	700	0.37

Tabela 6.2: Ocene modelov pri regresiji glede na aktivnost.

Model	#zn.	10xCV10	Malh.	u-t
(M11) unigrami+sw	600	49.9	70.1	52.8
(M12) trigrami+pos	900	40.7	64.5	40.7
(M1X4T) unigrami+st - tfidf	900	62.2	/	59.4
(M1X3T) unigrami - tfidf	900	57.8	/	58.5
(M21) #POS_tags	22	42.3	51.0	40.5
(M22) #Slang+ACL+FCL	3	38.3	36.7	31.4
(M23) #POS_tags - tfidf	27	43.6	/	41.7
(M31) Zn. strukture pesmi	9	36.6	34.7	32.1
(M42) DAL+ANEW	11	61.4	65.3	51.5
(M42X) DAL+ANEW - celotna	17	61.3	/	55.7
(M43) GI	125	62.8	61.7	56.9
(M43X) GI - tfidf	160	65.9	/	59.4
(M45) Warriner	17	59.7	/	52.8
C1Q (M11, M12)	800	51.56	69.9	49.16
C2Q (M21, M22)	17	39.80	52.9	30.22
C4Q (M42, M43)	57	59.62	76.2	47.91
C1234Q (C1Q, C2Q, M31, C4Q)	600	61.32	80.1	53.64
vsi modeli	600	70.37	/	58.43
vsi modeli - XGB	300	63.97	/	60.65

Tabela 6.3: Ocene modelov pri klasifikaciji v kvadrante.

Klasifikacija na podlagi regresije

Poskusili smo še, kako se obnese klasifikacija na podlagi naših regresorjev. Z regresorjem za valenco smo napovedali valenco čustev v besedilu in z regresorjem za aktivnost smo napovedali aktivnost čustev v besedilu. Na podlagi teh dveh vrednosti smo določili, v kateri kvadrant spada besedilo. Pri Malheiru [46] se je takšna klasifikacija izkazala (76.1%) za primerljivo mešanemu modelu C1234Q (80.1%). Mi smo s klasifikacijo na podlagi regresije na testnih podatkih dosegli oceno 34%.

6.2.2 Valenca

Preverili smo, kako dobro modeli diskriminirajo med poloblama glede na valenco. Primere, ki spadajo v kvadranta Q1 in Q2, smo označili kot pozitivne primere, za negativne primere pa primere, ki spadajo v kvadranta Q3 in Q4. V tabeli 6.4 so predstavljeni nekateri rezultati klasifikacije glede na valenco. Model z vsemi značilkami se dobro obnese, saj je tudi na testnih podatkih dosegel rezultat blizu 80%. Model, naučen z XGBoost, je dosegel podoben rezultat kot najboljši model, naučen s SVM.

6.2.3 Aktivnost

Preverili smo, kako dobro modeli diskriminirajo med poloblama glede na aktivnost. Primere, ki spadajo v kvadranta Q1 in Q4, smo označili kot pozitivne primere, za negativne primere pa primere, ki spadajo v kvadranta Q2 in Q3. V tabeli 6.5 so predstavljeni nekateri rezultati klasifikacije glede na aktivnost. Rezultati so pričakovano nekoliko slabši kot pa pri klasifikaciji glede na valenco. Najboljši model z vsemi značilkami je na testnih podatkih dosegel oceno 73%.

6.2.4 Kvadranti, eden proti vsem

Poskusili smo tudi, kako dobro modeli ločijo en kvadrant od drugih. Tak binarni problem smo zastavili tako, da so vsi primeri, ki spadajo v izbrani kvadrant (npr. Q1), pozitivni učni primeri, vsi ostali, ki ne spadajo v izbrani kvadrant, pa negativni. V tabelah 6.6, 6.7, 6.8 in 6.9 so rezultati modelov pri binarni klasifikaciji v posamezne kvadrante Q1, Q2, Q3 in Q4. Najboljši rezultat je za kvadrant Q2 (89%), za ostale kvadrante pa so si ocene podobne (77%, 78%, 76%). Dobili smo slabše rezultate kot Malheiro [46], ki je za kvadrante Q1, Q2, Q3 in Q4 dobil ocene 88.6%, 91.5%, 90.2% in 88.6%. Modeli, naučeni z XGBoost, so dosegli podobne rezultate kot najboljši modeli za posamezne probleme.

Model	#zn.	10xCV10	Malh.	u-t
(M11) unigrami+sw	600	74.9	/	62.5
(M12) trigrami+pos	100	74.5	/	58.5
(M13) bigrami+pos	800	70.6	73.9	61.5
(M14) unigrami+st+sw	400	77.3	81.7	65.6
(M15) bigrami - tfidf	500	58.8	68.2	52.9
(M1X3T) unigrami - tfidf	800	79.0	/	67.5
(M21) #POS_tags	12	68.0	/	60.6
(M22) #Slang+ACL+FCL	3	59.7	50.9	53.4
(M23) #POS_tags - tfidf	27	65.6	69.4	56.0
(M31) Zn. strukture pesmi	5	59.1	58.1	49.9
(M42) DAL+ANEW	11	84.5	82.8	72.1
(M42X) DAL+ANEW - celotna	16	82.9	/	77.0
(M43) GI	37	84.6	82.2	72.9
(M43X) GI - tfidf	90	86.1	/	76.2
(M44) Synesketch	3	73.0	/	62.4
(M45) Warriner	6	85.5	/	77.1
C1V (M11, M12, M13)	850	75.26	85.6	67.61
C2V (M21, M22)	10	64.34	71.0	58.29
C4V (M42, M43, M44)	9	85.25	86.7	72.18
C1234V (C1V, C2V, M31, C4V)	700	82.19	90.0	75.76
vsi modeli	400	87.91	/	79.98
vsi modeli - XGB	2000	86.97	/	77.48

Tabela 6.4: Ocene modelov pri klasifikaciji v poloble glede na valenco.

6.3 Komentar

V splošnem smo dosegli slabše rezultate kot Malheiro [46]. Pri semantičnih značilkah se verjetno nekoliko kaže odsotnost LIWC značilk in dejstvo, da smo slovarje za kvadrante zgradili povsem avtomatsko, medtem ko jih je Malheiro [46] zgradil pol-avtomatsko. Z ostalimi vrstami novih značilk smo

Model	#zn.	10xCV10	Malh.	u-t
(M11) unigrami+sw	200	72.8	79.9	64.9
(M12) trigrami+pos	100	66.9	83.9	63.2
(M13) bigrami+pos	100	68.9	77.7	65.2
(M1X4) unigrami+st	500	76.8	/	68.2
(M1X3T) unigrami - tfidf	500	76.7	/	66.4
(M21) #POS_tags	22	69.1	77.0	66.2
(M22) #Slang+ACL+FCL	3	69.6	71.3	58.2
(M23) #POS_tags - tfidf	30	65.3	/	65.6
(M31) Zn. strukture pesmi	5	69.5	79.9	61.3
(M42) DAL+ANEW	17	73.1	79.8	62.5
(M42X) DAL+ANEW - celotna	12	73.4	/	63.8
(M43) GI	142	81.7	78.8	70.0
(M43X) GI - tfidf	72	80.6	/	70.1
(M44) Synesketch	7	63.2	63.0	59.0
(M45) Warriner	15	71.2	/	65.7
C1A (M11, M12, M13)	1000	74.59	82.7	66.05
C2A (M21, M22)	28	69.98	75.4	59.08
C4A (M42, M43, M44)	50	73.19	76.2	69.81
C1234A (C1A, C2A, M31, C4A)	400	76.84	88.3	68.63
vsi modeli	400	79.68	/	73.05
vsi modeli - XGB	500	79.39	/	72.04

Tabela 6.5: Ocene modelov pri klasifikaciji v poloble glede na aktivnost.

dosegli večinoma podobne rezultate.

Za najobetavnejše so se izkazale nove semantične značilke. Semantične značilke so v naših poskusih velikokrat dosegle celo boljše rezultate kot značilke vsebine. Na splošno so najboljše rezultate dosegle kombinacije vseh značilk.

Algoritem XGBoost se ni izkazal za boljšega kot SVM. Treba pa je upoštevati, da je bilo z algoritmom XGBoost narejenih veliko manj poskusov in iteracij

Model	#zn.	10xCV10	u-t
(M1X4T) unigrami+st - tfidf	300	79.1	67.4
(M1X3T) unigrami - tfidf	300	77.1	61.8
(M14) unigrami+st+sw	300	69.1	63.7
(M21) #POS_tags	17	60.3	61.4
(M22) #Slang+ACL+FCL	3	64.1	50.7
(M23) #POS_tags - tfidf	22	59.0	59.8
(M31) Zn. strukture pesmi	4	56.6	54.6
(M42) DAL+ANEW	16	66.7	58.7
(M42X) DAL+ANEW - celotna	10	74.2	64.3
(M43) GI	37	75.1	62.9
(M43X) GI - tfidf	37	74.1	65.7
(M44) Synesketch	4	59.2	54.6
(M45) Warriner	15	73.4	62.1
C1Q1 (M14, M1X3T, M1X4T)	200	75.65	63.24
C2Q1 (M21, M22)	17	70.76	51.53
C4Q1 (M42X, M43X, M45)	30	73.89	66.35
C1234Q1 (C1Q1, C2Q1, M31, C4Q1)	300	80.10	67.39
vsi modeli	400	77.45	64.46
vsi modeli - XGB	600	72.23	63.57

Tabela 6.6: Ocene modelov pri klasifikaciji v kvadrant Q1.

optimiziranja parametrov. Algoritem smo učili na najboljših značilkah, ki jih je izbral algoritem ReliefF. Verjetno bi boljše rezultate dosegli, če bi XGBoost učili na vseh značilkah, vendar se je to izkazalo za časovno potratno.

Model	#zn	10xCV10	u-t
(M1X4) unigrami+st	200	77.1	72.4
(M14T) unigrami+st+sw - tfidf	100	75.5	72.8
(M1X3T) unigrami - tfidf	500	68.3	75.5
(M21) #POS_tags	20	63.1	64.2
(M22) #Slang+ACL+FCL	3	61.3	54.4
(M23) #POS_tags - tfidf	22	65.3	64.3
(M31) Zn. strukture pesmi	12	57.4	58.5
(M42) DAL+ANEW	10	84.4	76.4
(M42X) DAL+ANEW - celotna	15	81.5	77.3
(M43) GI	55	91.2	80.4
(M43X) GI - tfidf	125	92.0	79.9
(M44) Synesketch	4	67.4	71.1
(M45) Warriner	20	83.5	83.3
C1Q2 (M14T, M1X3T, M1X4)	300	68.60	72.75
C2Q2 (M21, M22)	10	56.73	53.55
C4Q2 (M42, M43, M45)	60	82.80	83.85
C1234Q2 (C1Q2, C2Q2, M31, C4Q2)	500	87.64	82.89
vsi modeli	100	88.87	85.43
vsi modeli - XGB	1500	83.79	83.17

Tabela 6.7: Ocene modelov pri klasifikaciji v kvadrant Q2.

Model	#zn.	10xCV10	u-t
(M11T) unigrami+sw - tfidf	200	74.12	63.93
(M14T) unigrami+st+sw - tfidf	400	73.78	64.93
(M1X4T) unigrami+st - tfidf	400	72.60	65.67
(M21) #POS_tags	72	80.35	63.21
(M22) #Slang+ACL+FCL	107	79.16	64.44
(M23) #POS_tags - tfidf	8	71.37	63.82
(M31) Zn. strukture pesmi	11	71.29	62.53
(M42) DAL+ANEW	16	67.14	57.24
(M42X) DAL+ANEW - celotna	25	67.05	62.47
(M43) GI	27	65.90	63.80
(M43X) GI - tfidf	5	65.09	58.11
(M44) Synesketch	3	61.87	58.21
(M45) Warriner	9	58.94	52.58
C1Q3 (M11T, M14T, M1X4T)	400	68.95	68.54
C2Q3 (M22, M23)	75	71.29	60.02
C4Q3 (M42X, M43)	30	74.12	64.86
C1234Q3 (C1Q3, C2Q3, M31, C4Q3)	200	70.76	69.40
vsi modeli	200	78.18	63.91
vsi modeli - XGB	300	73.21	69.20

Tabela 6.8: Ocene modelov pri klasifikaciji v kvadrant Q3.

Model	#zn.	10xCV10	u-t
(M11T) unigrami+sw - tfidf	600	80.0	62.5
(M14T) unigrami+st+sw - tfidf	500	78.3	61.2
(M1X4T) unigrami+st - tfidf	400	78.2	61.4
(M1X3T) unigrami - tfidf	400	72.9	64.4
(M21) #POS_tags	27	63.1	59.2
(M22) #Slang+ACL+FCL	3	55.9	51.8
(M23) #POS_tags - tfidf	22	59.2	53.2
(M31) Zn. strukture pesmi	6	57.1	50.7
(M42) DAL+ANEW	13	71.9	66.0
(M42X) DAL+ANEW - celotna	15	71.3	66.6
(M43) GI	20	71.7	66.1
(M43X) GI - tfidf	55	71.7	69.8
(M44) Synesketch	5	61.9	49.8
(M45) Warriner	15	70.1	64.5
C1Q4 (M11T, M14T, M1X3T, M1X4T)	300	72.69	64.20
C2Q4 (M21, M22)	10	56.53	56.34
C4Q4 (M42X, M43X, M45)	40	75.88	68.08
C1234Q4 (C1Q4, C2Q4, M31, C4Q4)	300	71.79	70.44
vsi modeli	500	76.46	70.46
vsi modeli - XGB	200	66.89	65.64

Tabela 6.9: Ocene modelov pri klasifikaciji v kvadrant Q4

Poglavje 7

Zaključek

V diplomskem delu smo uspešno implementirali obsežen celosten sistem za pridobitev, obdelavo in klasifikacijo glasbenih besedil. Sistem je sposoben pridobiti veliko število značilk iz glasbenih besedil. Na pridobljenih značilkah smo naučili vrsto modelov za klasificiranje in regresijo besedil, glede na valenco in aktivnost prevladujočih čustev. Potrdili smo rezultate Malheirove študije [46], da so nove značilke dober dodatek že obstoječim modelom za klasifikacijo besedil glede na čustva.

Čeprav je bil sistem zgrajen za delo z angleškimi besedili, mislimo, da je sistem vseeno zasnovan tako, da se da enostavno in v kratkem času preoblikovati tako, da je primeren za analizo besedil tudi v drugih jezikih. Večino sistema bi prilagodili drugemu jeziku že samo, če bi zamenjali angleške slovarje s slovarji za želen jezik.

Kljub temu da so nekateri rezultati slabši, kot jih je dosegel Malheiro [46], smo s svojim delom vseeno zadovoljni. Slabše rezultate lahko zagotovo deloma pripišemo odsotnosti značilk iz LIWC-ja, ki so se v izvorni študiji dobro izkazale. Izboljšati bi se dalo tudi še preprocesiranje besedil, predvsem pri algoritmih za prepoznavanje prevoda in refrena v besedilu.

Do boljših rezultatov bi morda lahko prišli še z več novimi značilkami ali pa z uporabo drugih učnih algoritmov. Splačalo bi se preizkusiti ansambelsko metodo strojnega učenja, imenovano „stacking“, kjer na predikcijah naučenih

modelov z logistično regresijo naučimo nov model.

Verjamemo, da bi se nasploh izboljšali rezultati, če bi imeli večjo učno množico. Zanimiv je predlog [71], da bi večjo zbirko glasbenih besedil zgradili na podlagi oznak s spletnih virov, kot so Last.fm[1] in AllMusic tako, da bi posameznim oznakam oz. kategorijam za razpoloženje s teh virov pripisali vrednosti za valenco in aktivnost. Tako bi lahko enostavneje pridobili veliko množico besedil, za katera bi imeli približne vrednosti za valenco in aktivnost.

[1] `https://www.last.fm/`

Literatura

[1] Judy I Alpert and Mark I Alpert. Music influences on mood and purchase intentions. *Psychology & Marketing*, 7(2):109–133, 1990.

[2] Saikat Basu, Jaybrata Chakraborty, Arnab Bag, and Md Aftabuddin. A review on emotion recognition using speech. In *Inventive Communication and Computational Technologies (ICICCT), 2017 International Conference on*, pages 109–114. IEEE, 2017.

[3] Definition of beat. Dosegljivo: `http://onlineslangdictionary.com/meaning-definition-of/beat`. [Dostopano 1. 9. 2017].

[4] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[5] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[6] Anne J Blood and Robert J Zatorre. Intensely pleasurable responses to music correlate with activity in brain regions implicated in reward and emotion. *Proceedings of the National Academy of Sciences*, 98(20):11818–11823, 2001.

[7] Gordon C Bruner. Music, mood, and marketing. *the Journal of marketing*, pages 94–104, 1990.

[8] Young Hwan Cho and Kong Joo Lee. Automatic affect recognition using natural language processing techniques and manually built affect lexicon. *IEICE transactions on information and systems*, 89(12):2964–2971, 2006.

[9] Wei Rong Chu, Richard Tzong-Han Tsai, Ying-Sian Wu, Hui-Hsin Wu, Hung-Yi Chen, and Jane Yung-jen Hsu. Lamp, a lyrics and audio mandopop dataset for music mood estimation: Dataset compilation, system construction, and testing. In *Technologies and Applications of Artificial Intelligence (TAAI), 2010 International Conference on*, pages 53–59. IEEE, 2010.

[10] Jarosław Cichosz and Krzysztof Slot. Emotion recognition in speech signal using emotion-extracting binary decision trees. *Proceedings of Affective Computing and Intelligent Interaction*, 2007.

[11] Ira Cohen, Ashutosh Garg, Thomas S Huang, et al. Emotion recognition from facial expressions using multilevel hmm. In *Neural information processing systems*, volume 2, 2000.

[12] Jeffrey F Cohn and Gary S Katz. Bimodal expression of emotion by face and voice. In *Proceedings of the sixth ACM international conference on Multimedia: Face/gesture recognition and their applications*, pages 41–44. ACM, 1998.

[13] Roddy Cowie, Ellen Douglas-Cowie, Nicolas Tsapatsoulis, George Votsis, Stefanos Kollias, Winfried Fellenz, and John G Taylor. Emotion recognition in human-computer interaction. *IEEE Signal processing magazine*, 18(1):32–80, 2001.

[14] Charles Darwin. *The expression of the emotions in man and animals*. Oxford University Press, USA, 1998.

[15] Liyanage C De Silva and Pei Chi Ng. Bimodal emotion recognition. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 332–335. IEEE, 2000.

[16] Tuomas Eerola and Jonna K Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music*, 39(1):18–49, 2011.

[17] P Ekman. Emotion in the human face . cambridge cambridgeshire, 1982.

[18] Paul Ekman and Wallace V Friesen. *Unmasking the face: A guide to recognizing emotions from facial clues*. Ishk, 2003.

[19] Emotion. Dosegljivo: `https://www.merriam-webster.com/dictionary/emotion`. [Dostopano 28. 8. 2017].

[20] Emotion classification. Dosegljivo: `https://en.wikipedia.org/wiki/Emotion_classification`. [Dostopano 28. 8. 2017].

[21] Donald Glowinski, Antonio Camurri, Gualtiero Volpe, Nele Dael, and Klaus Scherer. Technique for automatic emotion recognition by body gesture analysis. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, pages 1–6. IEEE, 2008.

[22] Primož Godec. *Nova podatkovna zbirka in evalvacija algoritmov za ocenjevanje razpolozenja v glasbi*. PhD thesis, Univerza v Ljubljani, 2014.

[23] Christian Gold, Martin Voracek, and Tony Wigram. Effects of music therapy for children and adolescents with psychopathology: a meta-analysis. *Journal of Child Psychology and Psychiatry*, 45(6):1054–1063, 2004.

[24] Didier Grandjean, David Sander, and Klaus R Scherer. Conscious emotional experience emerges as a function of multilevel, appraisal-driven

response synchronization. *Consciousness and cognition*, 17(2):484–495, 2008.

[25] Hatice Gunes. Automatic, dimensional and continuous emotion recognition. 2010.

[26] Andreas Haag, Silke Goronzy, Peter Schaich, and Jason Williams. Emotion recognition using bio-sensors: First steps towards an automatic system. In *Tutorial and research workshop on affective dialogue systems*, pages 36–48. Springer, 2004.

[27] Byeong-jun Han, Seungmin Ho, Roger B Dannenberg, and Eenjun Hwang. Smers: Music emotion recognition using support vector regression. 2009.

[28] Kun Han, Dong Yu, and Ivan Tashev. Speech emotion recognition using deep neural network and extreme learning machine. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[29] Toni Heittola, Anssi Klapuri, and Tuomas Virtanen. Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *ISMIR*, pages 327–332, 2009.

[30] Kate Hevner. Experimental studies of the elements of expression in music. *The American Journal of Psychology*, 48(2):246–268, 1936.

[31] Xiao Hu and J Stephen Downie. Exploring mood metadata: Relationships with genre, artist and usage metadata. In *ISMIR*, pages 67–72, 2007.

[32] Xiao Hu and J Stephen Downie. Improving mood classification in music digital libraries by combining lyrics and audio. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 159–168. ACM, 2010.

[33] Christian Martyn Jones and Tommy Troen. Biometric valence and arousal recognition. In *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, pages 191–194. ACM, 2007.

[34] Patrik N Juslin and John A Sloboda. *Music and emotion: Theory and research.* Oxford University Press, 2001.

[35] Patrik N Juslin and Marcel R Zentner. Current trends in the study of music and emotion: Overture. *Musicae scientiae*, 5(1_suppl):3–21, 2001.

[36] Youngmoo E Kim, Erik M Schmidt, Raymond Migneco, Brandon G Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A Speck, and Douglas Turnbull. Music emotion recognition: A state of the art review. In *Proc. ISMIR*, pages 255–266, 2010.

[37] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[38] Agata Kołakowska, Agnieszka Landowska, Mariusz Szwoch, Wioleta Szwoch, and Michal R Wrobel. Emotion recognition and its applications. In *Human-Computer Systems Interaction: Backgrounds and Applications 3*, pages 51–62. Springer, 2014.

[39] Igor Kononenko, Marko Robnik-Sikonja, and Uros Pompe. Relieff for estimation and discretization of attributes in classification, regression, and ilp problems. *Artificial intelligence: methodology, systems, applications*, pages 31–40, 1996.

[40] Cyril Laurier, Jens Grivolla, and Perfecto Herrera. Multimodal music mood classification using audio and lyrics. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*, pages 688–693. IEEE, 2008.

[41] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289. ACM, 2003.

[42] Yi-Lin Lin and Gang Wei. Speech emotion recognition based on hmm and svm. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 8, pages 4898–4901. IEEE, 2005.

[43] Yu-Ching Lin, Yi-Hsuan Yang, Homer H Chen, I-Bin Liao, and Yeh-Chin Ho. Exploiting genre for music emotion classification. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 618–621. IEEE, 2009.

[44] Cheng-Yu Lu, Jen-Shin Hong, and Samuel Cruz-Lara. Emotion detection in textual information by semantic role labeling and web mining techniques. In *Third Taiwanese-French Conference on Information Technology-TFIT 2006*, 2006.

[45] Ricardo Malheiro, Renato Panda, Paulo Gomes, and R Paiva. Music emotion recognition from lyrics: A comparative study. 6th International Workshop on Machine Learning and Music (MML13). Held in Conjunction with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPPKDD13), 2013.

[46] Ricardo Malheiro, Renato Panda, Paulo Gomes, and Rui Pedro Paiva. Emotionally-relevant features for classification and regression of music lyrics. *IEEE Transactions on Affective Computing*, 2016.

[47] Albert Mehrabian. Basic dimensions for a general psychological theory implications for personality, social, environmental, and developmental studies. 1980.

[48] Vinod Menon and Daniel J Levitin. The rewards of music listening: response and physiological connectivity of the mesolimbic system. *Neuroimage*, 28(1):175–184, 2005.

[49] Philipp Michel and Rana El Kaliouby. Real time facial expression recognition in video using support vector machines. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 258–264. ACM, 2003.

[50] Daniel Neiberg, Kjell Elenius, and Kornel Laskowski. Emotion recognition in spontaneous speech using gmms. In *Ninth International Conference on Spoken Language Processing*, 2006.

[51] Gerhard Nierhaus. *Algorithmic composition: paradigms of automated music generation*. Springer Science & Business Media, 2009.

[52] Tin Lay Nwe, Say Wei Foo, and Liyanage C De Silva. Speech emotion recognition using hidden markov models. *Speech communication*, 41(4):603–623, 2003.

[53] Yixiong Pan, Peipei Shen, and Liping Shen. Speech emotion recognition using support vector machine. *International Journal of Smart Home*, 6(2):101–108, 2012.

[54] Renato Panda, Ricardo Malheiro, Bruno Rocha, António Oliveira, and Rui Pedro Paiva. Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis. In *International Symposium on Computer Music Multidisciplinary Research*, 2013.

[55] Slav Petrov. Announcing syntaxnet: The world's most accurate parser goes open source. *Google Research Blog*, 2016.

[56] refren. Dosegljivo: `http://bos.zrc-sazu.si/cgi/a03.exe?expression=ge&name=sskj_testa`. [Dostopano 28. 8. 2017].

[57] James A Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178, 1980.

[58] Klaus R Scherer, Angela Schorr, and Tom Johnstone. *Appraisal processes in emotion: Theory, methods, research*. Oxford University Press, 2001.

[59] Klaus R Scherer and Marcel R Zentner. Emotional effects of music: Production rules. *Music and emotion: Theory and research*, pages 361–392, 2001.

[60] Erik M Schmidt and Youngmoo E Kim. Modeling musical emotion dynamics with conditional random fields. In *ISMIR*, pages 777–782. Miami (Florida), USA, 2011.

[61] Björn Schuller, Gerhard Rigoll, and Manfred Lang. Hidden markov model-based speech emotion recognition. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 1, pages I–401. IEEE, 2003.

[62] Abu Sayeed Md Sohail and Prabir Bhattacharya. Classification of facial expressions using k-nearest neighbor classifier. In *International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications*, pages 555–566. Springer, 2007.

[63] Yading Song, Simon Dixon, and Marcus Pearce. Evaluation of musical features for emotion classification. In *ISMIR*, pages 523–528, 2012.

[64] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The penn treebank: an overview. In *Treebanks*, pages 5–22. Springer, 2003.

[65] Robert E Thayer. *The biopsychology of mood and arousal*. Oxford University Press, 1990.

[66] Silvan Tomkins. *Affect imagery consciousness: Volume I: The positive affects*. Springer publishing company, 1962.

[67] Silvan Solomon Tomkins. Affect imagery consciousness, 2: The negative affects. 1963.

[68] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207, 2013.

[69] Chung-Hsien Wu, Ze-Jing Chuang, and Yu-Chung Lin. Emotion recognition from text using semantic labels and separable mixture models. *ACM transactions on Asian language information processing (TALIP)*, 5(2):165–183, 2006.

[70] Yi-Hsuan Yang and Homer H Chen. *Music emotion recognition*. CRC Press, 2011.

[71] Yi-Hsuan Yang and Homer H Chen. Machine recognition of music emotion: A review. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):40, 2012.

[72] Yi-Hsuan Yang, Yu-Ching Lin, Ya-Fan Su, and Homer H Chen. A regression approach to music emotion recognition. *IEEE Transactions on audio, speech, and language processing*, 16(2):448–457, 2008.

[73] Yasunari Yoshitomi, Sung-Ill Kim, Takako Kawano, and T Kilazoe. Effect of sensor fusion for recognition of emotional states using voice, face image and thermal image of face. In *Robot and Human Interactive Communication, 2000. RO-MAN 2000. Proceedings. 9th IEEE International Workshop on*, pages 178–183. IEEE, 2000.